Efficient, Concurrent Bayesian Analysis of Full Waveform LaDAR Data

Jing Ye

A thesis submitted for the degree of Doctor of Philosophy

Heriot-Watt University

School of Engineering and Physical Sciences
Department of Electrical, Electronic & Computer Engineering

October 2011

Acknowledgements

My deeply gratitude goes first and foremost to my supervisors Professor Andrew Wallace and Professor John Thompson, for their constant encouragement and guidance. They have supported me during the completion of my research with their consistent and illuminating instruction, whilst allowing me the room to work in my own way. Their penetrating and insightful comments facilitated me greatly through my thesis writing.

My sincere thank goes to Professor Gregory John Michaelson, for his kind concern and consideration regarding my academic requirements. I would like to express my heartfelt gratitude to Dr. Abyd Al Zain, for his continuous support, patience and steadfast encouragement to complete the project.

I should like to acknowledge the UK Engineering and Physical Sciences Research Council, Grant Reference EP/F030592, for their support under a research scholarship. It is my pleasure to thank the considerable help of Professor Gerald S. Buller, Dr. Aongus McCarthy and Nils Krichel in collecting the data necessary for the experiments reported in the thesis.

Most specially, I owe my sincerely thanks to my parents for their spiritual support and great confidence in me all through these years. I am also indebted to Sean who is always with me and gives me warm encouragement and love in every situation.

Abstract

Bayesian analysis of full waveform laser detection and ranging (LaDAR) signals using reversible jump Markov chain Monte Carlo (RJMCMC) algorithms have shown higher estimation accuracy, resolution and sensitivity to detect weak signatures for 3D surface profiling, and construct multiple layer images with varying number of surface returns. However, it is computational expensive. Although parallel computing has the potential to reduce both the processing time and the requirement for persistent memory storage, parallelizing the serial sampling procedure in RJMCMC is a significant challenge in both statistical and computing domains. While several strategies have been developed for Markov chain Monte Carlo (MCMC) parallelization, these are usually restricted to fixed dimensional parameter estimates, and not obviously applicable to RJMCMC for varying dimensional signal analysis.

In the statistical domain, we propose an effective, concurrent RJMCMC algorithm, state space decomposition RJMCMC (SSD-RJMCMC), which divides the entire state space into groups and assign to each an independent RJMCMC chain with restricted variation of model dimensions. It intrinsically has a parallel structure, a form of model-level parallelization. Applying the convergence diagnostic, we can adaptively assess the convergence of the Markov chain on-the-fly and so dynamically terminate the chain generation. Evaluations on both synthetic and real data demonstrate that the concurrent chains have shorter convergence length and hence improved sampling efficiency. Parallel exploration of the candidate models, in conjunction with an error detection and correction scheme, improves the reliability of surface detection. By adaptively generating a complimentary MCMC sequence for the determined model, it enhances the accuracy for surface profiling.

In the computing domain, we develop a data parallel SSD-RJMCMC (DP SSD-RJMCMC^{U}) to achieve efficient parallel implementation on a distributed computer cluster. Adding data-level parallelization on top of the model-level parallelization, it formalizes a task queue and introduces an automatic scheduler for dynamic task allocation. These two strategies successfully diminish the load imbalance that occurred in SSD-RJMCMC. Thanks to the coarse granularity, the processors communicate at a very low frequency. The MPI-based implementation on a Beowulf cluster demonstrates that compared with RJMCMC, DP SSD-RJMCMC^{U} has further reduced problem size and computation complexity. Therefore, it can achieve a super linear speedup if the number of data segments and processors are chosen wisely.

Glossary

ADC	Analogue to digital converter
AHW	Absolute half-width
AIC	Akaike information criterion
ALS	Airborne laser scanning
AM	Amplitude-modulated
BIC	Bayesian information criterion
CPU	Central processing unit
DP SSD-RJMCMC^{U}	Data parallel SSD-RJMCMC
DRA	Delayed rejection algorithm
EM	Expectation-maximization
FM	Frequency-modulated
GPU	Graphic processing unit
HWD	Heidelberger and Welch diagnostic
LaDAR	Laser detection and ranging
$(MC)^3$	Metropolis-Coupled MCMC
MCMC	Markov chain Monte Carlo
MDL	Minimum description length
M-H	Metropolis-Hastings
MLE	Maximum likelihood estimates
MIMD	Multiple instruction, multiple data
MISD	Multiple instruction, single data
MPI	Message passing interface
NSE	Numerical standard error
$P(MC)^3$	Parallel MC^3
PDF	Probability density function
PSRF	Potential scale reduction factor
RHW	Relative half-width

RJMCMC	Reversible jump Markov chain Monte Carlo
RNE	Relative numerical efficiency
SIMD	Single instruction, multiple data
SISD	Single instruction, single data
SPAD	Single-photon avalanche diode
SSD-RJMCMC	State space decomposition RJMCMC
TCSPC	Time-correlated single photon counting
TOF	Time-of-flight

Contents

List of Figures

List of Tables

Publications of the Candidate

- J. Ye, A. M. Wallace and J. Thompson, Parallel Markov Chain Monte Carlo Computation for Varying-dimension Signal Analysis, Proc. of the 17th European Signal Processing Conference (EUSIPCO 2009).
- A. M. Wallace, J. Ye, N. J. Krichel, A. McCarthy, R. J. Collins and G. S. Buller, Full Waveform Analysis for Long-Range 3D Imaging Laser Radar, EURASIP Journal on Advances in Signal Processing, Volume 2010, Article ID 896708.
- N. J. Krichel, A. McCarthy, A. M. Wallace, J. Ye, and G. S. Buller, Long-range Depth Imaging Using Time-correlated Single-photon Counting, Proc. of SPIE - The International Society for Optical Engineering, Volume 7780, 77801I (2010).
- J. Ye, A. M. Wallace, J. Thompson, An Adaptive, Concurrent Reversible Jump Markov Chain Monte Carlo Method with Application to LaDAR Signal Analysis, submitted to Journal of Computational Statistics & Data Analysis.
- J. Ye, A. M. Wallace, A. Al. Zain, J. Thompson, Parallel Bayesian Inference of Range and Intensity from LaDAR Profiles, submitted to Journal of Parallel and Distributed Computing.

Chapter 1

Introduction

1.1 Background and Overview

In view of their ability to capture scene geometry directly, rather than infer such information from passive data, active range data capture systems have become prevalent for 3D imaging and surface profiling. In particular, there is an increasing requirement for *non-contact* 3D data acquisition techniques, having a wide range of applications in the industrial, medical, remote sensing and defence areas. For example, this includes target detection, recognition and classification with eye-safe [1] device power consumption for defence and security, 3D surface profile characterization for metrology in the automotive and airborne industries, and object scanning and mapping for archaeological sites.

Our laboratory has developed a powerful optical remote sensing technology, a laser detection and ranging (LaDAR) system, for range profiling and surface characterization based on a low-power pulsed laser source and time-correlated single photon counting (TCSPC) detector. This technology has significant advantages in better depth resolution, higher

[1] Eye-safe means the power consumption does cause eye injuries. If eyes are not protected adequately when working with high-power device such as laser beams, severe damage can occur.

measurement accuracy and sensitivity to low returns.

The objective for LaDAR data interpretation is to retrieve a complete 3D surface geometry viewed by the laser imaging system. Conventionally, deterministic methods can only deal with a single target return from an opaque surface with high reflectivity. Recently, considerable interest has grown in the situation where the received signals are composed of multiple surface reflections within the laser footprint. This can occur in several application areas, for example in defence and security when imaging remote, often camouflaged targets, or in remote aerial sensing, when a wide footprint often results in multiple returns, e.g. from a forest canopy. Statistically, the problem can be reduced to inference about a parameter vector whose dimensionality is not fixed. Researchers have proposed several signal processing techniques within the frequentist framework [1]. Although these methods are effective in many cases, they fail to resolve two closely separated surfaces, and are not able to produce satisfactory results when the background level is comparable or higher than the signal amplitudes.

In order to detect multiple, small returns embedded in background, noise and clutter, we have been developing a modern paradigm within the Bayesian framework using reversible-jump Markov chain Monte Carlo (RJMCMC) sampling algorithms. Earlier work has shown RJMCMC can lead to dramatically improved estimation accuracy, resolution and sensitivity. However, the long computation cycle and the large memory storage requirement largely limit its feasibility.

In recent years, parallel computing has received impetus due to the increasing availability of cheap computing power and networking. The basic idea is to divide a large computation task into several smaller ones, and execute them concurrently on separate processors. This offers the possibility to cope with massive scientific computation and complex statistical

[1] Frequentist inference is to formulate generally applicable schemes for making statistical inferences, i.e. for drawing conclusions from statistical samples . The well-established methodologies are statistical hypothesis testing and confidence intervals.

problems based on intensive numerical algorithms. Within 3D image capture, it has the potential to conquer the restrictions on significant computation time and large memory storage in RJMCMC sampling.

However, parallel processing of RJMCMC algorithms is not only a technical implementation challenge, but also an open statistical problem. Since a Markov chain is serial by nature, RJMCMC cannot be directly migrated onto a parallel system. The difficulty is to make use of the conditional independence structure of the underlying model and divide the sampling procedure into independent tasks. Although there have been several strategies for Markov chain Monte Carlo (MCMC) parallelization, they are originally designed for parameter estimation with fixed dimensionality, and are not obviously applicable to an RJMCMC algorithm such as the varying dimensional problem in LaDAR. Where the existing algorithms could be adapted for RJMCMC, there is a danger of degrading the simulation efficiency by introducing additional undesired calculation or implementation complexity.

Therefore, our motivation is to improve the simulation efficiency of LaDAR signal analysis, first by parallelization of an RJMCMC algorithm, second by adopting a strategy that adapts to the data, and third by developing an effective implementation on a distributed cluster computer.

1.2 Summary of Contributions

This thesis reports several contributions to the state-of-the-art knowledge in the discipline of efficient, concurrent Bayesian inference of full waveform LaDAR signals using RJMCMC algorithms.

First, we have investigated the use of convergence diagnostics to determine the proper

length of the Markov chain. RJMCMC sequences are random trials; their convergence performance is data dependent; and only a finite number of samples can be generated in practical applications. By assessing the mixing performance of the sampler output, the diagnostic can effectively detect and remove the Markov transient period, evaluate whether the constructed sequence has converged to its limiting distribution of interest, and dynamically terminate the Markov process.

Second, we have implemented and evaluated two parallel RJMCMC strategies for LaDAR signal processing. One is a direct and simple approach, parallelizing complex calculations, which introduces a massive message passing overhead and degrades the simulation efficiency. The other is to construct multiple independent MCMC chains to explore and compare the candidate models in RJMCMC, where no communication occurs during the chain generation. Although for some other applications, MCMC sequences can have better mixing performance than trans-model RJMCMC, resulting in a faster convergence rate and increased simulation efficiency, there are convergence problems when processing LaDAR data observing a comparatively large number of surface returns.

Third, we have proposed a variant of RJMCMC, namely state space decomposition RJMCMC (SSD-RJMCMC), to exploit model-level parallelization by decomposing and reconfiguring the state space of RJMCMC. The designed framework draws on the complementary advantages of the RJMCMC and parallel MCMC methods, and avoids their drawbacks. By sharing the between-model difficulty among a set of independent concurrent RJMCMC sequences with restricted variation of model dimensionality (i.e. the surface return), the convergence length is considerably shorter than standard RJMCMC. Moreover, the re-configured state space guarantees a complete exploration over the entire state space, addressing the local optimal problem in standard RJMCMC. Furthermore, the embedded error detection and correction scheme improves the reliability of Bayesian model selection (i.e. the inference of surface number). Finally, based on the convergence performance of model related parameters, we can adaptively generate a complimentary

MCMC sequence for the found model to enhance the accuracy and precision of parameter estimates (i.e. surface profiling).

Fourth, we have developed an efficient implementation of SSD-RJMCMC on a distributed computer cluster, called data parallel SSD-RJMCMC (DP SSD-RJMCMC^{U}). Adding data-level parallelization on top of the model-level parallelization, DP SSD-RJMCMC^{U} inherits the statistical merits of SSD-RJMCMC but conquers the challenges for parallel implementation. Thanks to the coarse granularity, the processors communicate at a very low frequency. With the formalized task queue and the intelligent scheduler for dynamic task allocation, it solves the load-balancing problem in SSD-RJMCMC and adapts the implementation to a flexible size of clusters.

1.3 Thesis Outline

The structure of the thesis is as follows:

Chapter 2 reviews existing 3D imaging systems and introduces one laser ranging technique of particular interest to this thesis, Time-Correlated Single Photon Counting (TCSPC). After summarizing some related LaDAR processing approaches and identifying their limitations, we introduce the Bayesian paradigm and Bayesian modelling of LaDAR. Subsequently, we describe how to apply Markov chain Monte Carlo (MCMC) and reversible jump Markov chain Monte Carlo (RJMCMC) algorithms to conduct Bayesian inference of parameters from a LaDAR signal with known and unknown numbers of surface returns respectively. Due to the requirement of dynamic chain length control, we review current convergence diagnostics with discussions on their advantages and disadvantages, and select those best suited to LaDAR. Finally, we demonstrate the benefit our algorithm on real data sets with single and multiple returns. This chapter addresses the statistical concepts for Bayesian analysis of LaDAR signals, upon which we develop the

parallel techniques in following three chapters.

Chapter 3 discusses the inefficiency of RJMCMC which directs our research to parallel processing. We briefly introduce the relevant background within parallel computing and discuss in detail the practical considerations of RJMCMC parallelization in both statistical and computing domains. We then review and classify the current parallel strategies for MCMC algorithms with their benefits and drawbacks. After that, we implement a direct and simple approach safely applicable to RJMCMC, the parallel likelihood computation method. Finally, we present the proposed solution for varying-dimensional RJMCMC parallelism, the parallel MCMC chains method, and compare the performance to the serial implementation of a benchmark, coal mining disaster problem. The analytical tractability for LaDAR problem, caused by the converge difficulty, motivates us and provides with the guidance to develop a proper RJMCMC parallelization strategy in the next chapter.

Chapter 4 presents the new algorithm, SSD-RJMCMC, for RJMCMC parallelism, borrowing the idea of parallel exploration from the parallel MCMC chains method and retaining the between-model jumps in RJMCMC. The algorithm is illustrated on a synthetic data set. Comparing with standard RJMCMC, we demonstrate the drawbacks of RJMCMC, and highlight the advantages of SSD-RJMCMC in model selection and parameter estimation. The performance is evaluated on a real data set with multiple surface returns with known surface geometry. This chapter addresses the parallelization difficulty in the statistical domain, while the next chapter develops the parallel implementation with regard to the parallel computing domain.

Chapter 5 addresses the practical challenges of SSD-RJMCMC for high level speedup and develops an effective strategy, DP SSD-RJMCMC^{U}, to accomplish an efficient implementation on a Beowulf cluster with distributed memories. We start with SSD-RJMCMC and arrange the processing steps to reduce inter-processor communication overhead. The method is demonstrated on both synthetic and real LaDAR data with multiple surface re-

turns. We compare the proposed approach with standard RJMCMC and single processor SSD-RJMCMC, showing the speedup and efficiency improvements.

Chapter 6 concludes the thesis with suggestions for future work.

Chapter 2

Bayesian Analysis of Full Waveform LaDAR Signals using RJMCMC Algorithms

2.1 Full Waveform 3D LaDAR Imaging

2.1.1 Background of 3D Imaging

Broadly speaking, we classify the optical range-finding techniques into two categories according to their operational principle: *passive* and *active* 3D imaging systems. Practical passive systems rely on the acquisition and processing of two or more intensity images acquired from different viewpoints. Provided the corresponding points or features can be identified within the two or more images, it is possible to recover the 3D information from the scene using the laws of projective (or other) geometry. The identification of corresponding points and 3D mapping process are based on the calibration of the optical system and the structure of the scene. For a calibrated system with full knowledge of the intrinsic camera and extrinsic placement parameters, 3D mapping is a simple pro-

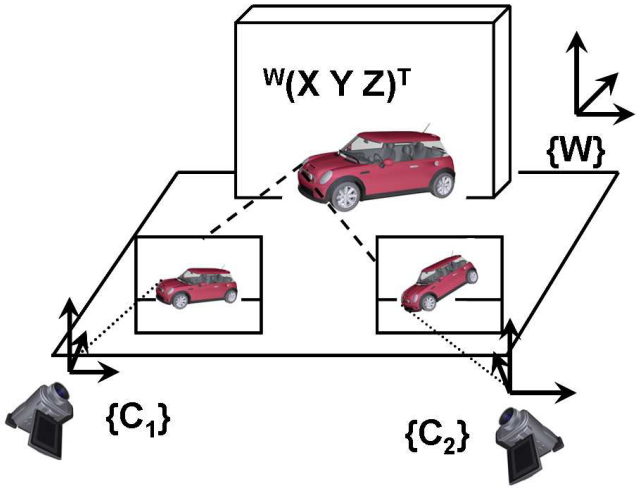


Figure 2.1: An example of passive optical system: passive stereo principle. It uses two or more cameras to observe image features, such as edges or points, and matches the features on an object based on the intersect of two lines of sight through the same feature point on each camera's image plane.

cess. In recent years, researchers have made significant progress to retrieve 3D data using uncalibrated system [3, 4].

Although passive techniques are useful in the scenarios where the artificial energy sources of radiations are prevented, they can observe limitations for 3D imaging. For example, due to the fundamental problems of defocus and image blur, passive systems are rarely used for the *focusing methods* [5, 6]. Because of the close dependency on the illumination and the surface reflectance property of the object scene, the *shape-from-X* techniques [7], where X can be shading, motion, texture, silhouette, etc., are not appropriate for 3D depth data acquisition. The methods of *(passive) stereo* [8, 9] (see Figure 2.1) are computationally complicated and are not suitable to deal with the non-cooperative targets since no metric data is provided outside highly constrained scenarios.

Active 3D imaging systems use the projection of an energy source, usually a laser, but also microwave radar, millimetre waves or sonar, onto the scene, and then acquire one or more

images of the scene subject to the energy coverage, such as the laser illumination for a laser source. Different from the passive system, active systems eliminate the identification and matching of the corresponding salient features, and allow the direct acquisition of 3D data. In general, these methods are much more robust and accurate than passive techniques.

The most common method of active laser imaging is the laser detection and ranging (LaDAR) technique [10], also known as light detection and ranging (LiDAR). LaDAR systems show a large dynamic range, from a few metres to several kilometres, and measurement uncertainties of signal strength related with the reflectivity of the target. In practice, the active laser source and the receiver are closely located, which facilitates a compact set up and avoids the occlusion problem in previous techniques.

LaDAR system generates the angle-angle-range 3D images, where the (x, y) coordinates are computed from the angular resolution using the direction of the transmitted laser signal, and the distance to the target (z) is from range measurement for each pixel. Basically, there are two techniques for range measurement, triangulation and time-of-flight (TOF). The triangulation system [11] makes use of at least two known scene viewpoints and forms a triangle from the laser spot, the camera and laser emitter. The TOF system acquires the absolute 3D measurement along the line of sight of a common transmitter-receiver optical axis, which immediately eliminates the occlusion problems in the triangulation systems. In the works presented in this thesis, we consider the TOF system, which will be described in details in the subsequent section.

2.1.2 Time-of-flight LaDAR System

As illustrated in Figure 2.2, in the TOF LaDAR system, the basic principle for an isolate imaging element is to project a laser signal towards the object surface and acquire the re-

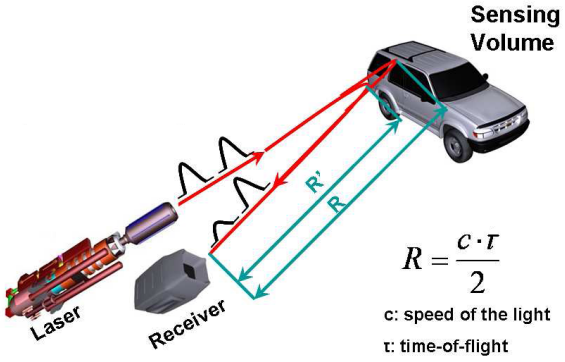


Figure 2.2: TOF LaDAR principle.

flection by a receiving optical system. The round trip time delay between the laser pulse emission and the returned signals is measured, and the actual distance to the surface is determined by half of the product of the light velocity (c) and the total traveling time (τ), that is $\frac{c \cdot \tau}{2}$. To construct the full 3D image, it is necessary to employ a scanning mechanism either in two dimensions with a single detector or in one dimension with a linear detector array [12, 13]; otherwise, a focal plane array [14, 15] or micro-channel plates [16] can be used to acquire concurrent pixel detail. While arrayed detectors provide parallel data acquisition, which has clear advantages in acquiring data from moving targets and eliminating scanning components, there are problems with crosstalk and fill-factor. In general, we can achieve better temporal response and sensitivity with a single element detector, which is of considerable importance for covert, low-power operation. Therefore, in this thesis, we are concerned with the single element detector.

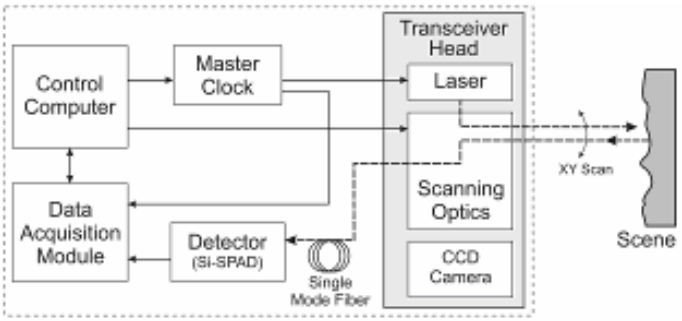
In order to measure the time of flight, two different techniques can be applied, the analogue ramp and the laser beam modulation. In the former method, a proportion of the laser pulse is diverted to a photodiode or other detector. This start signal is amplified and used to trigger a ramp, which is the basis of the time interval measurement. The returning signal from the target is detected by the same or another detector to stop the ramp. The ramp amplitude provides a measurement of the elapsed time and hence the distance to the

target. The most common solutions in laser beam modulation are amplitude-modulated (AM) TOF and frequency-modulated (FM) TOF. The AM technique [17] modulates the beam sinusoidally at a fixed frequency and measures the phase shift between the transmitted and received signal, which provides the equivalent measures of the time. The FM [18] measures the output power that is a function of the modulation frequency and the distance to target.

There are a number of important factors in maintaining the timing resolution and hence distance accuracy in TOF ranging system, including the shape (rise time, full width half maximum) of the laser pulse, and the efficiency and response time of the photon detector. For the signal conditioning and measurement, the linearity, or at least the calibrated repeatability of the ramp, minimization of the timing jitter and the resolution of the analogue to digital converter(ADC) and the digital timing circuit must all be optimized. Assuming these considerations can be satisfied, the accuracy of these TOF systems is dependent on receiving a relatively strong signal from the target than can be detected and proposed. This is largely influenced by the reflectance properties and the range of the target. Additionally, the systems are temperature sensitive, although this can be alleviated either by temperature control or by use of a reference channel for calibration. These limitations have led us to develop the TOF imaging system based on time-correlated single photon counting (TCSPC) technique.

2.1.3 Time-of-Flight LaDAR System using Time-correlated Single Photon Counting

Time-correlated single photon counting (TCSPC) is a statistical sampling technique with single photon detection sensitivity, capable of sub-picoseconds timing resolution. In general, as shown in Figure 2.3 picoseconds-duration laser pulse is directed towards a non-cooperative target, and the scattered photon returns trigger a single-photon detector sys-



(a)

(b)

Figure 2.3: (a) Schematic diagram indicating the principal components of the scanning system. Electrical paths are denoted by solid lines, optical paths by dashed lines. Si-SPAD is a silicon single-photon avalanche diode. (b) The transceiver head assembly. The system dimensions are approximately 275mm by 275mm by 175mm. The two galvanometer servo-control circuit boards (not visible) are on the underside of the slotted baseplate.

tem. Each photon return can be regarded as an independent measurement of the round traveling time, and the high repetition rate pulsed laser source yields a collection of multiple photon returns, of which the measured return time is displayed as a histogram.

The system uses a pulsed semiconductor diode laser, of pulse half-width 90ps, operating at 842nm wavelength, that emits low energy pulses (<30pJ). The laser is capable of operating at repetitions rates in excess of 10MHz, although 2MHz was the maximum rate used in our measurements. Scene scanning is performed by a pair of galvanometer mirrors. The optical system is used to direct the outgoing laser pulses onto each optical field position of the target, and also to efficiently collect the scattered photons returned from each corresponding pixel of the imaged scene. The collected return photons are routed using polarisation optics to an individual, high performance single-photon detector module via a single mode optical fiber. The signal from the single-photon detector is recorded as a timed photon event, equivalent to range (z) which can be associated with an (x, y) coordinate that is known from the calibrated scanning optics. For the particular optical configuration and scanning parameters used in our measurements, the maximum field of view was 55mrad and the beam width and scanning resolution were both approximately 23mm at a stand-off distance at 325m.

In general, each detector event records a photon arrival, some of which are returned from the target, some from stray events (other light sources), and some will be due to detector, dark counts. To reduce the stray photon events, our system includes spatial filtering (by coupling into the single mode optical fiber), spectral filtering (by narrowband filtering at the known laser wavelength) and temporal filtering (by the TCSPC technique, as there is finite window in which to record a photon event). Timing uncertainty is introduced by jitter in the master clock, the laser driver, the detector (silicon SPAD) and the timing electronics. For all those reasons, we use many pulses to build up a statistical distribution of the number of recorded photon arrivals as a function of the arrival time. This can be interpreted as a range measurement, and by scanning and recording distributions at each

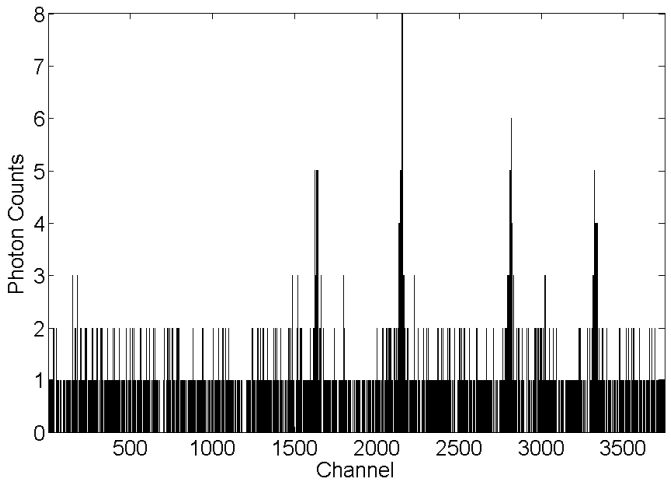


Figure 2.4: Multiple returns recorded from a distributed target in the field of view of a single pixel. The horizontal axis is equivalent to the round-trip distance, and the vertical axis a measure of the strength of signal return.

pixel, as a depth image. An example of a measurement that records data from more than one surface in the field of view of a single pixel is shown in Figure 2.4.

Compared with previous methods for 3D ranging and imaging based on time-of-flight, TCSPC system offers several benefits. First, the repeating and averaging of the time measurement considerably improve the time resolution and make it shorter than the system jitter, which is typically tens of picoseconds. The accurate time measurement supports a higher depth resolution and a reconstruction of the surface details. Second, the highly repetitive characteristic is ideal for the low scattered surfaces, and enhances the sensibility of the system to cope with variation in amplitude of the reflected return of several orders of magnitude. Third, it is also desirable that the average power of the active laser pulse in photon-counting technology is sufficiently low for eye-safe and low-light level operation, an attractive factor in many defence and security applications.

2.1.4 Related Work for LaDAR Signal Analysis

The twin demands of low power and multi-waveform analysis of TOF TCSPC signals place significant demands on the signal processing methodology. In the simplest case, it is possible to use the straightforward centroid method to obtain the range estimates. The temporal separation (and hence the range) of the surface returns is obtained by thresholding a prominent signal, for example, at a fixed percentage of the peak amplitude, and computing the centroid of the thresholded data. On the one hand, the advantage of this approach is that it is nonparametric, i.e., it does not require the knowledge of the instrumental response (the shape of the return), which makes it a significantly efficient method. On the other hand, this restricts its application to the signals with high signal-to-background ratio. Nevertheless, in real applications, signals can be very weak since the surfaces can be distant and of poor reflectivity.

If the instrumental response is known, a possible solution is to apply a matched filter in advance to improve the signal-to-background ratio. However, the previous matched filtering of the histogram can still fail to distinguish surface returns with comparable or even lower amplitude than the background level. Moreover, this method cannot resolve the closely separated surfaces, since the centroid computation can only be displaced at the intermediate channels bins of the merged returns. For these reason, more sophisticated approaches should be developed to provide more accurate solutions, such as the statistical algorithms.

Typical techniques within the frequentist framework are to calculate the Maximum Likelihood Estimates (MLE) of parameters for every possible number of signal returns, and then use information theoretic criteria, such as Akaike (AIC), Bayesian Information Criterion (BIC) and Minimum Description Length (MDL) [19], to determine the signal number. One popular tool for finding MLE is Expectation-Maximization (EM) [20]. Compared with centroid method and matched filter, this algorithm is computationally more inten-

sive, but it may give estimates of higher accuracy. However, EM holds a potential risk in that it might converge to a local maximum likelihood or diverge to an infinite value [21]. Additionally, it is sensitive to initial values and not efficient for data set containing numerous observed events, in our case the timing information for the received photons. Moreover, even though AIC, BIC and MDL introduce penalty terms to avoid over fitting the data, that is adding more returns to increase the likelihood, they still have the tendency to produce more complicated models which correspond to more signal returns [22].

Wallace *et al.* [23] proposed a hybrid approach, which first applies a deterministic non-parametric bump-hunting process for initial estimates of signal returns, and second Poisson-MLE to refine the estimates. Although it is effective in many cases, it is not able to produce satisfactory results when the number of returns increases or the returned signals are occluded severely by the background noise.

Advanced algorithms within the Bayesian framework inferring the posterior distribution of the signal parameters are proved to be more successful in providing more accurate and reliable estimates for the mixture models, for instance, the multiple surface returns in LaDAR signals. Normally, the posterior density is of a complicated form, which is almost analytically intractable. The practical solution is to construct a random process which draws samples from the target posterior distribution. The commonly used sampling approaches are Markov chain Monte Carlo (MCMC) algorithms for fixed dimensional parameter analysis, and the reversible jump Markov chain Monte Carlo (RJMCMC) algorithms for varying dimensional parameter analysis. Starting from this idea, Hernandez-Marin *et al.* [22] develops a Bayesian statistical approach based on RJMCMC techniques. When the remote target involves an unknown number of surfaces, the posterior is defined over a countable number of parameter subspaces, with each subspace corresponding to a fixed number of returns. The generated samples can both jump from one subspace to another via birth/death/split/merge moves to assess the number of the surface returns, and explore within the subspaces to update the position and amplitude of the fixed number of

returns. The main advantage of this algorithm is that it incorporates uncertainty in both the model and the prior specification, which exposes multiple explanations of the set of plausible reconstructions and avoids the unsatisfactory and difficult hypothesis testing for the number of surface returns. The practical examples demonstrate that this approach can provide a higher degree of accuracy for the depth imaging than previous techniques. However, this algorithm suffers from between-model mixing difficulty and computationally intensive. Designing the data-driven or non-uniform between-model move kernels might help to explore the state space more efficiently, but they can introduce additional algorithmic complexity and sophisticated computation.

There are some alternative approaches for trans-dimensional simulation that also make use of the Markov chain Monte Carlo algorithms to explore the posterior distribution for the Bayesian analysis but have not yet been applied to LaDAR processing. Grenander and Miller [24] proposed a sampling strategy known as jump-diffusion. It comprises two move types: the discontinuous jumps between subspaces with different dimensionality, and the diffusion process within the subspaces using the Langevin stochastic differential equation. The diffusion process reshapes the template of the subspace by performing the statistical drift following the gradients of the posterior energy. The difficulty of jump-diffusion dynamic is to establish the statistical differential equation and define the associated transformations. In addition, the drift terms and the derivatives can be computationally more expensive than the likelihood computation in the RJMCMC.

Stephens [25] proposed the birth-death MCMC to address the varying dimensional simulation. In this work, the different parameters of the mixture model are sampled from the marked point process by constructing a continuous time Markov birth-death process. One realization of the marked point process [26, 27] consists of a set of the isolated point either in time or geographical space, where in image analysis, each point can be associated with a mark specifying the geometric property of the underlying object. Stephens claims that this continuous-time simulation is simpler to implement than RJMCMC. However, Cappe

et al. [28] argues the simplicity will be lost if more complicate moves are incorporated in the between-model moves such as split and merge in LaDAR analysis.

Different from TCSPC LaDAR data discussed above, where each temporal response (i.e. each surface return) is a statistical average of the accumulated backscattered photon counts, the airborne laser scanning (ALS) technique interprets the temporal distortion of the backscattered energy profile. Each of the energy waveform (called an echo) represents an individual reflection from an object and the echo shape provides the physical characteristics of the illuminated surfaces. Decomposition of the echoes allows the reconstruction of the 3D target range. Classical methods to resolve the echoes are wavelets, spines [29] and Gaussian mixture models [30]. However, they are not able to cope with different echo shape and therefore loss the physical property. The non-linear least-square approach using the gradient computation [31] has the same limitations on physical knowledge. More recently, Mallet [32] investigates a marked point process approach applying the RJMCMC sampler coupled with a simulated annealing to ensure and accelerate the convergence to the desired probability measure. It takes into account of both the echo shapes in signals and the related parametric functions, and therefore can successfully reconstruct the signal and meanwhile retrieve the physical information.

2.2 Bayesian Analysis of Full Waveform LaDAR Signal using RJMCMC Algorithms

Statistical inference plays a significant role in computer vision and image processing, such as targets detection and tracking, image restoration, 3D reconstruction and so forth. To conduct the statistical inference, we need to firstly formulate a statistical model that can adequately describe the situation of interest, and then quantify the uncertainty and extract the conclusions from the observation using either the frequentist approach or Bayesian ap-

proach. As discussed in Section 2.1.4, in the multiple surface return scenarios, frequentist approach may fail to detect the small surface return or separate the merged returns from the complicated target response waveform. Therefore, the more complex algorithms in Bayesian framework are required to reliably determine the number and ranging of target surfaces.

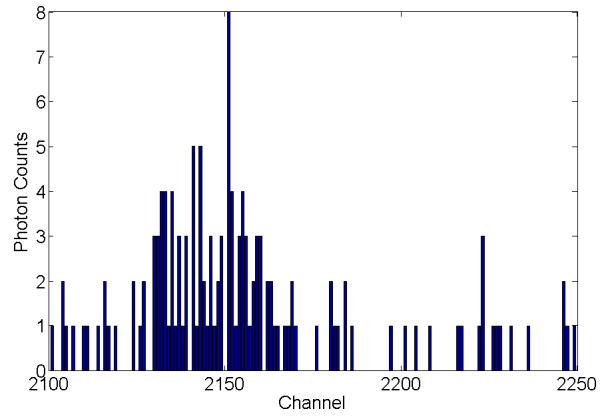
2.2.1 LaDAR Signal Modelling

As presented in Section 2.1.3, the system response generated by the repeat laser pulses is in the form of a histogram of accumulated photon counts at different round-trip times. Due to the footprint of the beam impinging on a target with surfaces distributed in depth or with semi-transparency, the laser return can observe multiple peaks, where each peak stands for one surface return, peak position indicates the target range and peak amplitude is related with surface reflectivity.

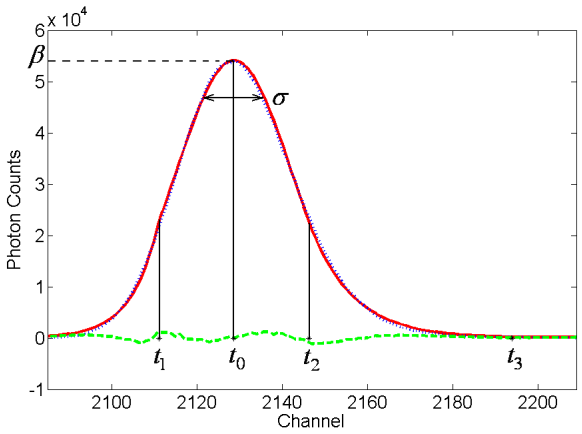
To interpret the data, we employ the piecewise exponential function to model the instrumental response of a single peak, which was originally introduced in [33]. The parametric form of the expected temporal variation of the photon count distribution in channel i is given by

$$f_{\text{system}}(i;\beta,t_0) = \beta \begin{cases} e^{\frac{-(t_1-t_0)^2}{2\sigma^2}} e^{\frac{(i-t_1)}{\tau_1}}, & i < t_1 \\ e^{\frac{-(i-t_0)^2}{2\sigma^2}}, & t_1 \leq i < t_2 \\ e^{\frac{-(t_2-t_0)^2}{2\sigma^2}} e^{\frac{-(i-t_2)}{\tau_2}}, & t_2 \leq i < t_3 \\ e^{\frac{-(t_2-t_0)^2}{2\sigma^2}} e^{\frac{-(t_3-t_2)}{\tau_2}} e^{\frac{-(i-t_3)}{\tau_3}}, & i \geq t_3 \end{cases} \tag{2.1}$$

where β is an amplitude factor, t_0 is the time of the peak maximum, and t_1, t_2 and t_3 are the points at which the changeovers between functions occur (see Figure 2.5). In the methods we have developed, the shape parameters are assumed to be fixed and known from the instrumental response. Hence, our prime objective for each single peak is to fit



(a)



(b)

Figure 2.5: (a) A single magnified selected peak. (b) Instrumental response of TCSPC LaDAR signal (dotted line) and fitting result (solid line) using piecewise exponential model with fitting errors (dashed line). The parameter sets corresponding to Equation (2.1) are: $\beta = 5.41 \times 4$, $t_0 = 2128.70$, (t_1, t_2, t_3) =(2111.15,2146.36,2193.93), (τ_1, τ_2, τ_3) =(6.32,10.04,292.79).

the received signal with the instrumental response, in other words, to estimate the peak amplitude β and peak position t_0.

In the general case where there are multiple returns, the histogram **y** is a combination of the overlapped peaks against a background level, whose expected value is assumed to be constant across all the channel bins. It can be considered as a sample of a non-normalized

statistical mixture distribution with density $F(i;k,\phi)$, defined as

$$F(i;k,\phi) = \sum_{j=1}^{k} f_{\text{system}}(i;\beta_j, t_{0_j}) + B \tag{2.2}$$

where k is the number of peaks, B is the background and ϕ is the parameter set subject to inference: $\phi = (\beta, t_0, B)$ with $\beta = (\beta_1, \beta_2, ..., \beta_k)$ and $t_0 = (t_{0_1}, t_{0_2}, ..., t_{0_k})$.

Assuming the observations in each channel are conditionally independent given the model parameters, the histogram is considered as a discrete representation of a spatially heterogeneous Poisson process whose intensity is a linear superposition of the scaled and shifted returns defined in Equation (2.1). Accordingly, the number of photon counts y_i recorded in each channel i of the histogram is a random sample of a Poisson distribution with probability

$$P(y_i|k,\phi) = e^{-F(i;k,\phi)} \frac{F(i;k,\phi)^{y_i}}{y_i!}. \tag{2.3}$$

The joint probability distribution of the entire histogram **y** is

$$L(\mathbf{y}|k,\phi) = \prod_{i=1}^{i_{max}} e^{-F(i;k,\phi)} \frac{F(i;k,\phi)^{y_i}}{y_i!}. \tag{2.4}$$

2.2.2 Bayesian Inference for LaDAR

2.2.2.1 Bayesian Approach

In Bayesian inference, the uncertainties associated with the unknowns are expressed in forms of probability distribution which are updated from the observed evidence. The underlying theory is to estimate the posterior distribution of the hypothesis (parameter vector θ) given the observation (**y**), using the combination of the prior probability over the hypothesis and the likelihood of the evidence. The posterior distribution is a con-

ditional distribution that is assigned after taking into account of relevant evidence; the prior distribution expresses the uncertainty about the hypothesis without any evident; and the likelihood integrates the compatibility of the observed evidence with the hypothesis. Compared with with frequentist approaches, the distinct advantage of Bayesian computation is that it invokes a natural method to take fuller account of the uncertainty related with the statistical model and parameter values by incorporating the prior knowledge when updating the hypothesis in response to the new information.

The mathematical formulation of Bayesian inference is defined through Bayes theorem:

$$\pi(\theta|\mathbf{y}) = \frac{\pi(\theta)L(\mathbf{y}|\theta)}{\int \pi(\theta')L(\mathbf{y}|\theta')d(\theta')} \tag{2.5}$$

where $\pi(\theta)L(\mathbf{y}|\theta)$ is the joint probability distribution of the random quantities $\mathbf{y}$ and θ, $L(\mathbf{y}|\theta)$ the likelihood function, $\pi(\theta)$ the prior distribution on the parameter θ and $\int \pi(\theta')L(\mathbf{y}|\theta')d(\theta')$ is a normalising constant known as the marginal density of $\mathbf{y}$.

Once the posterior $\pi(\theta|\mathbf{y})$ is obtained, any features of the posterior distribution are in accordance with Bayesian inference, such as moments, quantiles and highest posterior density regions. All these quantities can be express in the form of the posterior expectation of functions of θ:

$$E[f(\theta)|\mathbf{y}] = \frac{\int f(\theta)\pi(\theta)L(\mathbf{y}|\theta)d(\theta)}{\int \pi(\theta)\pi(\mathbf{y}|\theta)d(\theta)} \tag{2.6}$$

The Bayesian approach is particularly attractive for the model selection problem, which is to make a discrete choice between a set of models with different dimensionality, and estimate the parameter vector depending on the model in question based on the observed evidence. Some examples are variable selection in regression, object recognition, mixture deconvolution with an unknown number of parameters, to give but a few examples. All such problems can be formulated as a matter of joint inference about a model indicator k and a parameter vector ϕ_k of dimension n_k. In the Bayesian paradigm, this is the inference

of a joint posterior $\pi(k, \phi_k|\mathbf{y})$, which can be factorised as the product of the posterior model probabilities ($\pi(k|\mathbf{y})$) and the model-specific parameter posteriors ($\pi(\phi_k|k, \mathbf{y})$),

$$\pi(k, \phi_k|\mathbf{y}) = \pi(k|\mathbf{y})\pi(\phi_k|k, \mathbf{y}) \tag{2.7}$$

2.2.2.2 Bayesian Modelling of LaDAR Data

The objective for full waveform LaDAR analysis, as stated in Section 2.2.1, is the inference about the unknowns k and ϕ so as to capture the accurate information of the number of peaks, the peak positions, peak amplitudes, and the background level of the returned signal. In Bayesian framework, the problem is to estimate the joint posterior distribution $\pi(k, \phi|\mathbf{y})$ of (k, ϕ) from the histogram of photon counts. This is formulated as:

$$\pi(k, \phi|\mathbf{y}) = \frac{L(k, \phi|\mathbf{y})\mathbf{f}(k, \phi)}{\int L(k, \phi|\mathbf{y})\mathbf{f}(k, \phi)\delta(k, \phi)} \propto L(k, \phi|\mathbf{y})\mathbf{f}(k, \phi). \tag{2.8}$$

where $L(k, \phi|\mathbf{y})$ is the likelihood function defined in Equation (2.4) and $\mathbf{f}(k, \phi)$ is the full joint prior distribution.

The unknowns are assumed *prior* independent and the full joint prior distribution $\mathbf{f}(k, \phi)$ can be modelled as the product of each individual prior:

$$\mathbf{f}(k, \phi) = \text{prior}(k) \times \prod_{j=1}^{k} \text{prior}(t_{0_j}) \times \prod_{j=1}^{k} \text{prior}(\beta_j) \times \text{prior}(B) \tag{2.9}$$

whereas the number of peaks follows a uniform distribution on $[0, k_{\text{max}}]$, with k_{max} chosen to be suitably large. The position t_{0_j} is drawn from a uniform distribution with support in the interval $[0, i_{\text{max}}]$, assuming no prior knowledge of peak positions. For the prior distribution of β_j and B, we assume a gamma distribution, which is a natural choice in statistics community for parameters defined in $\Re^+$. Accordingly, Equation (2.9) is

specified as:

$$\mathbf{f}(k,\phi) = \frac{1}{k_{\max}} \times (\frac{1}{i_{\max}})^k \times \prod_{j=1}^{k} f_G(\beta_j|a,b) \times f_G(B|c,d) \tag{2.10}$$

where f_G is the probability density function (PDF) of gamma distribution with shape parameters a, c and scale parameters b, d respectively.

Normally, the target distribution defined in Equation (2.8) is intractable and difficult to analysis. Therefore, we consider using the powerful simulation algorithms, such as MCMC and RJMCMC, to sample from the posterior distribution and conclude the estimation from the sample trajectories.

2.2.3 MCMC Methodology with Application to LaDAR

As expressed in Equation (2.6), to make inference about model parameters and related statistical features, Bayesians need to integrate over the probability distributions with possibly high-dimensions, as those arising when dealing with LaDAR data. However, in most applications, analytical evaluation of $E[f(\theta)|\mathbf{y}]$ is difficult and complicate. The alternative is to draw sample from the target distribution, i.e. the model-specific parameter posteriors $\pi(\phi_k|k,\mathbf{y})$ in Bayesian framework, and evaluate the statistical properties from the generated samples. In this section, we introduce two commonly used sampling approaches: Monte Carlo method and Markov chain Monte Carlo (MCMC) algorithms, and then describe the MCMC sampler design for the LaDAR problem.

2.2.3.1 Monte Carlo Methods

To avoid an unnecessarily Bayesian flavour in the following discussion, we comprise model parameters θ and observed data $\mathbf{y}$ as X, and restate the Equation (2.6) in a more

general term:

$$E[f(X)] = \frac{\int f(x)\pi(x)dx}{\int \pi(x)dx} \tag{2.11}$$

Monte Carlo methods approximate the population mean of $f(X)$ as a sample mean, using a number of samples $\{X_t, t = 1, ?n\}$ drawn from $\pi(\cdot)$:

$$E[f(X)] \approx \frac{1}{n}\sum_{t=1}^{n} f(X_t). \tag{2.12}$$

When the samples are independent, the law of large numbers ensures that the approximations can become as accurate as desired by increasing the number of simulated values. In general, when $\pi(\cdot)$ is complex and not a standard distribution, it is infeasible to draw independent samples $\{X_t\}$ from $\pi(\cdot)$. However, $\{X_t\}$ need not necessarily be independent. They can be generated by any process with the support of $\pi(\cdot)$, where a Markov chain is one way of doing this by setting $\pi(\cdot)$ as its limiting, invariant target distribution.

2.2.3.2 Markov Chains

Markov chain is a memoryless random process. It generates a sequence of random variables $\{X_t, t = 1, ..., n\}$, wherein the next state X_{t+1} at time $t + 1$ depends only on the current state X_t of the chain, and does not further depend on the history of the chain $\{X_0, X_1, ..., X_{t-1}\}$, formally:

$$Pr(X_{t+1} = x | X_1 = x_1, X_2 = x_2, ..., X_n = x_n) = Pr(X_{t+1} = x | X_n = x_n) \tag{2.13}$$

X_{t+1} is sampled from a transition probability $Pr(X_{t+1}|X_t)$. We assume the chain is time-homogeneous when transition probability is independent of time t. The evolution of the

Markov chain on space $\Omega \subseteq \Re$ is governed by the *transition kernel* [34], defined as:

$$P(x, A) = Pr(X_{t+1} \in A | X_t = x), x \in \Omega, A \subset \Omega \tag{2.14}$$

Generally, the transition kernel is a combination of both the continuous and discrete components. Suppose for some $p(x, y) : \Omega \times \Omega \rightarrow \Re^+$, the kernel can be expressed as:

$$P(x, dy) = p(x, y)dy + r(x)\delta_x(dy) \tag{2.15}$$

where $p(x, x) = 0, \delta_x(dy) = 1$ if $x \in dy$ and 0; otherwise, $r(x) = 1 - \int_\Omega p(x, y)dy$. This kernel means the transitions from x to y with probability $p(x, y)$ and x to x with probability $r(x)$.

Basically, the goal of the analysis is to specify conditions under which the constructed Markov chain converges to the invariant distribution as $t \rightarrow \infty$, and the averages of the sample trajectories satisfy the law of large numbers and the central limit theorem. To that, we need to introduce some related definitions and properties.

Invariant: For a Markov chain on a space $\Omega \subseteq \Re^+$, the invariant distribution π^* is the one that satisfies

$$\pi^*(dy) = \int_\Omega P(x, dy)\pi(x)dx \tag{2.16}$$

where $\pi(x)$ is the density of π^* with respect to Lebesgue measure.

Reversible: A Markov chain is said to be reversible if it satisfies the *detailed-balance* condition

$$\pi(dx)P(x, dy) = \pi(dy)P(y, dx) \tag{2.17}$$

The reversibility condition is stronger than Equation (2.16) to obtain an invariant distribution. We evaluate the right and left side of Equation (2.16), and the invariance can be

verified:

$$\begin{aligned}\int P(x,A)\pi(x)dx &= \int\{\int_A p(x,y)dy\}\pi(x)dx + \int r(x)\delta_x(A)\pi(x)dx \\ &= \int_A\{\int p(x,y)\pi(x)dx\}dy + \int_A r(x)\pi(x)dx \\ &= \int_A\{\int p(y,x)\pi(y)dx\}dy + \int_A r(x)\pi(x)dx \\ &= \int_A(1-r(y))\pi(y)dy + \int_A r(x)\pi(x)dx \\ &= \int_A \pi(y)dy \end{aligned} \tag{2.18}$$

Irreducible: Irreducibility means the Markov chain is able to visit all the sets in the state space Ω with the positive probability under $\pi(\cdot)^*$ from any starting point within a finite number of iterations. Formally, a Markov chain is said to be π^*-irreducible if for every $x \in \Omega$

$$\pi^*(A) > 0 \Rightarrow P(X_t \in A | X_0 = x_0) > 0, A \subset \Omega \tag{2.19}$$

Aperiodic: Aperiodicity ensures that the chain does not cycle through a finite number of sets. A Markov chain is said to be aperiodic if there is no partition of $\Omega = (D_0, D_1, ..., D_m)$ for some $m > 2$ such that for $\forall t$

$$P(X_t \in D_{(t)\text{mod(m)}} | X_0 \in D_0) = 1 \tag{2.20}$$

These definitions allow us to state the following results from [35] which links the Markov chain on continuous state space and the theory for Markov chain Monte Carlo methods.

Theorem 1 *Suppose $\{X_t, t = 1, ..., n\}$ is a π^*-irreducible Markov chain with transition kernel $P(\cdot,\cdot)$ and invariant distribution π^*, then π^* is the unique invariant distribution of*

$P(\cdot,\cdot)$ and for all π^*-integrable real-valued functions h,

$$\frac{1}{M}\sum_{t=1}^{M} h(X_t) \rightarrow \int h(y)\pi(dy) \text{ as } M\rightarrow\infty \tag{2.21}$$

Theorem 2 *Suppose $\{X_t, t = 1, ..., n\}$ is a π^*-irreducible, aperiodic Markov chain with transition kernel $P(\cdot,\cdot)$ and invariant distribution π^*. Then for π^*-almost every $x \in \Omega$, and all sets A*

$$\|P^M(x, A) - \pi^*(A)\| \rightarrow 0 \text{ as } M\rightarrow\infty \tag{2.22}$$

where $\|\cdot\|$ denotes the total variant distance.

The first theorem gives conditions under which a strong law of large numbers holds. It guarantees the chain will gradually forget its initial states and will eventually converge to the unique stationary (invariant) distribution, irrespective of t or X_0. As t increases, the samples X_t will increasingly look like the dependent samples from π^*. After a sufficient long transient period (burn-in) from the initial state, the output from the Markov chain can be used to estimate the expectation $E[f(x)]$. Discarding the m samples in burn-in period, the remaining samples gives an estimator, which is an *ergodic average*:

$$\overline{f} = \frac{1}{n-m}\sum_{t=m+1}^{n} f(X_t) \tag{2.23}$$

The second theorem gives conditions under which the probability density of the M^{th} iterate of the Markov chain converges to its unique, invariant density. To obtain a central limit theorem, a further crucial requirement to strengthen the conditions is that the chain needs to be ergodic, i.e., the chains must be irreducible, aperiodic and Harris-recurrent, wherein the latter ensures the same limiting behavior for every starting value (see [35] for definition). The central limit theorem holds for all the initial distributions if it holds for the invariant distribution.

2.2.3.3 Markov Chain Monte Carlo Methods

Equation (2.23) shows that $E[f(x)]$ can be estimated from the correlated samples in the Markov chain, where the expectation is taken over its stationary distribution $\pi(\cdot)$. This provides a solution for the Bayesian inference of parameters in LaDAR signals, but first we need to construct a Markov chain, and then ensure its stationary distribution is our distribution of interest. Two simplest recipes to construct such a Markov chain are the Metropolis-Hastings (M-H) algorithm [36] and a Gibbs sampling [37].

Metropolis-Hastings Algorithms

In Metropolis-Hastings algorithms, each updating step of the Markov chain is divided into two part: a proposal and an acceptance of the proposal. The proposal suggests a random state for the next step in the sample trajectory, and the acceptance ensures the convergence to the target distribution. The sampling procedure is:

- Initialize x_0 and set $t = 0$
- Repeat:
  1. Sample x' from $q(\cdot, x_t)$
  2. Sample u from a uniform distribution $U(0, 1)$
  3. If $u \leq \alpha(x_t, x')$, accept x' and set $x_{t+1} = x'$; otherwise set $x_{t+1} = x_t$.

The acceptance probability, $\alpha(x, x')$, is derived and interpreted in [38] . As proved in Equation (2.18), a Markov chain will converge to the invariant distribution if it is reversible by satisfying the detailed balance condition, and the unique invariant distribution is in support of a unique transition kernel $P(\cdot, \cdot)$. Since the transition kernel can be complicated, it is difficult to sample directly from $P(\cdot, \cdot)$. For this reason, we choose an arbitrary proposal distribution $q(\cdot, \cdot)$, usually in form of a standard function. However, q is not likely to be reversible for π. Without loss of generality, with $\pi(x)q(x, x') > \pi(x')q(x', x)$, the rate of transition from x to x' exceeds that in the reverse direction. To

balance the transition between x and x', we introduce a function $0 \leq \alpha(x, x') \leq 1$, such that $\pi(x)q(x, x')\alpha(x, x') = \pi(x')q(x', x)$. This defines the two components of q in M-H sampling:

$$p(x, x') = q(x, x')\alpha(x, x'), x \neq x' \tag{2.24}$$

Accordingly, by satisfying the following condition, the M-H kernels has π as its invariant density:

$$q(x, x')\alpha(x, x')\pi(x) = q(x', x)\alpha(x', x)\pi(x') \tag{2.25}$$

It immediately gives the final expression of $\alpha(x, x')$:

$$\alpha(x, x') = \min\{1, \frac{\pi(x')q(x', x)}{\pi(x)q(x, x')}\} \tag{2.26}$$

Gibbs Sampler

Suppose we have a joint distribution $p(X_1, ..., X_k)$ that we need to sample from (for example, a posterior distribution). If we knew the *full conditional distributions* for each parameter, we can use Gibbs sampler to generate a sequence of samples from that joint distribution. For each parameter, the full conditional distribution is the distribution of the parameter conditional on the known information and all the other parameters: $p(X_j|X_{-j}, \mathbf{y})$. The Gibbs sampler updates the parameter component one by one, drawing samples from the full conditional distributions in turn:

Given $x^{(t)} = (x_1^{(t)}, ..., x_m^{(t)})$, generate
1. $X_1^{t+1} \sim p(x_1|x_2^{(t)}, ..., x_m^{(t)})$
2. $X_2^{t+1} \sim p(x_2|x_1^{(t+1)}, x_3^{(t)}, ..., x_m^{(t)})$
...
m. $X_m^{t+1} \sim p(x_m|x_1^{(t+1)}, ..., x_{m-1}^{(t+1)})$

The Gibbs sampler is, formally, equivalent to the m single-element Metropolis-Hastings algorithms with acceptance probability equal to 1 and proposal distribution equal to the

full conditional distribution. However, it implies limitations compared to the Metropolis-Hastings algorithm, since some (or all) of the full conditionals may not look like any distributions we know. Moreover, the Gibbs sampler is not applicable to varying dimensional parameter space, because the irreducibility condition on such space does not hold. Therefore, as Metropolis-Hastings algorithm suits LaDAR analysis with varying number of peak returns, and is much more convenient to implement, we follow the M-H kernels in this thesis.

2.2.3.4 MCMC Applied to LaDAR

For LaDAR signals with a fixed number of peaks (i.e. k is known), we can define the following move types and the transition kernels, which satisfies the aperiodic, irreducible and invariant property such that the Markov chain will converge towards the target limiting distribution. For each of the iteration (giving one sample), three moves are carried out consequently.

1. Update the positions t_0 using a random walk proposal:

$$q(t'_0|t_0) = \prod_{j=1}^{k} q_N(t'_{0_j}|t_{0_j}, \sigma_{t_0}) \tag{2.27}$$

   where q_N denotes a normal distribution density with mean and standard deviation to be the current position t_{0_j} and σ_{t_0} respectively.

2. Update the amplitudes β using a random walk proposal with variance σ_β:

$$q(\beta'|\beta) = \prod_{j=1}^{k} q_N(\beta'_j|\beta_j, \sigma_\beta) \tag{2.28}$$

3. Update the background B using a random walk proposal with variance σ_B:

$$q(B'|B) = q_N(B'|B, \sigma_B) \tag{2.29}$$

2.2.4 RJMCMC Methodology with Application to LaDAR

2.2.4.1 RJMCMC Methodology

MCMC algorithms are the direct approach to approximate the probability density of the parameter vector with fixed dimensionality, which is the model-specific parameter posteriors $\pi(\phi_k|k, \mathbf{y})$ in Bayesian model determination problems as defined in Equation (2.7). To enable inference of the joint posterior $\pi(k, \phi_k|\mathbf{y})$, we need to construct a single Markov chain with the state of (k, ϕ_k). This is called trans-dimensional simulation, where the state space is $\bigcup_{k\in\mathcal{K}}(\{k\} \times \Re^{n_k})$, with model k in the countable set $\mathcal{K}$. Green [39] proposed a reversible jump Markov chain Monte Carlo (RJMCMC) algorithm for the trans-dimension simulation, which is an extension of Metropolis-Hastings algorithm that does not only allow the within-model parameter updates for a particular k, but also the jumps between models with different k. Before describing the RJMCMC sampling scheme in detail, we first introduce a constructive representation of Metropolis-Hastings algorithm, which forms the basis of simulation in a variable dimension context [40].

In the MCMC chain on state space $\mathcal{X} \subset \Re$, suppose the current state x, the next state x' is proposed from a deterministic function $x' = h(x, u)$, where u, with dimension r, is the random number from a known joint density g. The reverse transition from x' to x is made using a random number u' from probability density g'. If the transformation from (x, u) to (x', u') is a diffeomorphism (the transformation and its inverse are differentiable),

Equation (2.25) can be written as the following $(d+r)$-dimensional equality:

$$\pi(x)g(u)\alpha(x,x') = \pi(x')g'(u')\alpha(x',x)|\frac{\partial(x',u')}{\partial(x,u)}| \tag{2.30}$$

where $|\frac{\partial(x',u')}{\partial(x,u)}|$ is the Jacobian term from the standard change-of-variable formula. Accordingly, α is derived as:

$$\alpha(x,x') = \min\{1, \frac{\pi(x')g'(u')}{\pi(x)g(u)}|\frac{\partial(x',u')}{\partial(x,u)}|\} \tag{2.31}$$

In this constructive representation, suppose the dimensions of x, x', u and u' are d, d', r and r', respectively, then for mapping $x' = h(x,u)$ and $x = h'(x,u)$, we have the deterministic functions $h : \Re^d \times \Re^r \to \Re^{d'}$ and $h' : \Re^{d'} \times \Re^{r'} \to \Re^d$. Since the transformation from (x,u) to (x',u') is diffeomorphism, the dimension must satisfy $d + r = d' + r'$; otherwise the mapping and its inverse could not be both differentiable. This is called "dimension-matching" between (x,u) and (x',u'). However, the individual dimensions of x and x' do not necessarily have to be equal, which means the expression (2.31) applies, without change, in a variable dimension context.

This "invisible" dimension-jump from x to x', of the same or different dimensions, provides a solution to our generic model deterministic problem, where we wish to use these reversible jump moves to sample the space $\bigcup_{k\in\mathcal{K}}(\{k\} \times \Re^{n_k})$ with invariant distribution $\pi(k,\phi_k|y)$. This is where the reversible jump Markov chain Monte Carlo algorithm comes from.

In RJMCMC, we still follow the Metropolis-Hastings sampling procedure as defined in the ordinary MCMC, and propose samples $x' = (k', \phi'_{k'})$ from $x = (k,\phi_k)$ in accordance with different move type m in a countable set $\mathcal{M}$. For each move type, the detailed balance

equation (2.25) becomes:

$$q_m(x, x')\alpha_m(x, x')\pi(x) = q_m(x', x)\alpha(x')\pi(x') \tag{2.32}$$

where $q_m(x, x')$ is the joint proposal distribution of move type m and destination x'. Based on (2.31), the acceptance probability of move type m is replaced by:

$$\alpha(x, x') = \min\{1, \frac{\pi(x')}{\pi(x)} \frac{j_m(x')}{j_m(x)} \frac{g'_m(u')}{g_m(u)} |\frac{\partial(x', u')}{\partial(x, u)}|\} \tag{2.33}$$

with $j_m(x)$ to be the probability of choosing move type m when at state x. The corresponding variables x, x', u and u' are of dimensions d_m, d'_m, r_m and r'_m, respectively, with the dimension-matching condition $d_m + r_m = d'_m + r'_m$, we have $x' = h_m(x, u)$ and $x_m = h'_m(x', u')$, where raises the Jacobian term related with move type m.

The combination of each move, with a reversible transition kernel in support of π, gives an ergodic chain, and the ergodic average as expressed in (2.23) can be used to select the model and infer the related model parameters.

2.2.4.2 RJMCMC Applied to LaDAR

If the number of peaks remains unknown, we need to introduce another four trans-model move types in the RJMCMC sampler. Within one iteration (called a "sweep"), one of the following four moves are performed following the within-model updating in MCMC.

- Within-model updating: update t_0, β and B.
- Trans-model updating:
    1. Random Birth or Death of a peak: The choice of either creating a new peak or deleting an existing peak is randomly made with probability b_k and d_k. If a

death move, we make a random selection over the current peaks, while for the birth move, the position and amplitude for the new generated peak is drawn from the uniform and gamma distribution respectively:

$$\beta_{k+1} \sim \Gamma(1, \frac{1}{i_{max}} \sum_{i=1}^{i_{max}} y_i) \tag{2.34}$$

$$t_{0_{k+1}} \sim U(0, i_{max}) \tag{2.35}$$

2. Random Splitting an existing peak into two peaks or Merging of two peaks into a single peak: We make a random choice of splitting or merging with probability s_k and m_k. Note the move probabilities need to satisfy the condition of $b_k + d_k + s_k + m_k = 1$. For splitting, we randomly choose a k_n which is to be split into two new peaks k'_n and k''_n with parameters (β'_n, t'_{0_n}) and (β''_n, t''_{0_n}) defined by the deterministic functions based on dimension matching. For the merging case, two adjacent peaks are randomly chosen and the new peak is constructed as:

$$\beta_n = \beta'_n + \beta''_n \tag{2.36}$$

$$t_{0_n} = \frac{t'_{0_n} + t''_{0_n}}{2} \tag{2.37}$$

The pseudo-code the RJMCMC is as follows:

```
Repeat sample generation
  if <current state contains at least one peak>
    // within-model update
    position update
    amplitude update
    background update
    // between-model move
    if <k = upper limit>
```

```
9       Death
10    else
11      randomly select a move type: Birth/Death/Split/Merge
12      perform the selecte between-model move
13  else
14    background update
15    Birth
```

2.2.5 Computational Complexity

An important issue for the Markov chain Monte Carlo simulation is the algorithm efficiency. Efficiency can be reflected by physical measures, for example, the time, memory, disk and the network usage, which depend on the machine, compiler and the code. Advances in computer technology make physical measures irrelevant. A more standardized measure is the computational complexity, which describe the resource requirements as a function of algorithm scale.

Some algorithms perform the same number of operations which consume constant time regardless of the problem scale. We are usually interested in the algorithms which conduct different number of operations depending on the problem size, for instance, the size of the input data. Computational complexity studies how the number of operations relates to the problem size. In particular, the main concern is the worst case, i.e. the upper bound of the operation elements for a given problem size. This is expressed by big-O notation (see for instance Sipser [41] for more information).

Big-O notation:
A function $T(N)$ is $O(F(N))$ if for some constant c and for all values of N greater than some value n_0:

$$T(N) \leq c \times F(N) \tag{2.38}$$

where $T(N)$ is the *exact* complexity of the algorithm as a function of problem size N, and $F(N)$ gives the *worst case* complexity.

To measure the complexity of MCMC/RJMCMC algorithms, we decompose each move type into a series of statements and outline the corresponding complexity using big-O notation. The big-O notation expresses how the algorithm complexity relates to the input N. It does not contain constant terms, as they become insignificant when N gets large enough. For this reason, we only identify the complexity affected by the problem size. The size of the problem is determined by (L, k, N), respectively the histogram length, the peak number and the length of the Markov chain. The complexity for different moves are summarized in Table 2.1.

```
// position update
  draw k position proposals --> O(k)
  compute instrumental response of each peak --> O(kL)
  compute log-likelihood value --> O(L)
  compute acceptance probability
  if <proposals are accepted>
    update the current state
  else
    remain in the current state
```

```
// amplitude update
  draw k amplitude proposals --> O(k)
  compute instrumental response of each peak --> O(kL)
  compute log-likelihood value --> O(L)
  compute prior probability --> O(L)
  compute acceptance probability
  if <proposals are accepted>
    update the current state
  else
    remain in the current state
```

```
// background update
```

```
  draw background proposal
  compute log-likelihood value --> O(L)
  compute acceptance probability
  if <proposal is accepted>
    update the current state
  else
    remain in the current state
```

```
// Birth
  propose a new peak
  compute instrumental response of the new peak --> O(L)
  compute log-likelihood value --> O(L)
  compute proposal probability
  compute prior probability of the new peak
  compute jump probability
  compute acceptance probability
  if <Birth is accepted>
    update the current state
  else
    remain in the current state
```

```
// Death
  random select an existing peak to delete
  compute log-likelihood value --> O(L)
  compute proposal probability
  compute prior probability of the deleted peak
  compute jump probability
  compute acceptance probability
  if <Death is accepted>
    update the current state
  else
    remain in the current state
```

```
// Split
  random select an existing peak to be split
```

```
  split the existing peak to two new peaks
  compute instrumental reponse of the new peaks --> O(L)
  compute log-likelihood value --> O(L)
  compute proposal probability
  compute prior probability
  compute jump probability
  compute Jacobian term
  compute acceptance probability
  if <Split is accepted>
    update the current state
  else
    remain in the current state
```

```
// Merge
  random select two peaks to be merged
  compute instrumental response of the merged peak --> O(L)
  compute log-likelihood value --> O(L)
  compute proposal probability
  compute prior probability
  compute jump probability
  compute Jacobian term
  compute acceptance probability
  if <Merge is accepted>
    update the current state
  else
    remain in the current state
```

2.3 Delayed Rejection Algorithm with Application to LaDAR

In the Metropolis-Hastings kernels for both practical MCMC and RJMCMC, a new value is generated from a proposal distribution of convenience, and then accepted or rejected

Move type	Computational complexity
Position update	$O(NkL)$
Amplitude update	$O(NkL)$
Background update	$O(NL)$
Birth	$O(NL)$
Death	$O(NL)$
Split	$O(NL)$
Merge	$O(NL)$

Table 2.1: Computational complexity of different move types in RJMCMC sampling using big-O notation.

in accordance with the acceptance probability with respect to π. The rejection of the candidate moves intrinsically ensures the detailed balance, and therefore preserves the invariant distribution. However, remaining in the current state intuitively increases the autocorrelation in the realized chain, which indicates that a proposal distribution is badly calibrated to the target distribution.

Tierney and Mira [42] exploits the idea of Peskun [43] that the performance of the sampler can be improved by reducing the probability of remaining in the current states, and designed the delayed rejection algorithm (DRA). Unlike the standard Metropolis-Hastings updates, when the candidate value (x') is rejected, instead of retaining the current value (x) and proceeding to the next move, the delayed rejection algorithm makes another attempt (x'') from a different proposal distribution, and then makes an accept/reject decision for the second candidate according to an adjusted acceptance probability. The obvious advantage of DRA is that it can reduce the autocorrelation of the constructed Markov chain and improve the efficiency of sample averaging by increasing the acceptance rate. Moreover, the candidates in the following stages can be proposed based on the previously rejected value, from which we can abstract the information and adaptively generates the new values to explore the state space more efficiently. Furthermore, this process is not limited to two stages and can be expanded as required. However, the process should only be repeated when the reduction in autocorrelation can compensate for the additional computation cost.

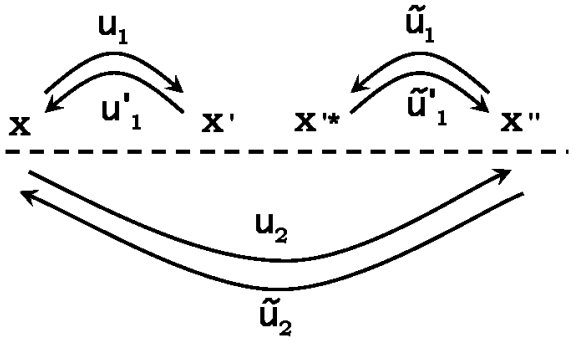


Figure 2.6: Schematic diagram adapted from [1] for delayed rejection algorithm.

In the Tierney and Mira's method, although learning is from the previously rejected value, the Markov property is not destroyed and all the Markov chain theories in MCMC still apply. In particular, the reversibility holds on the condition that the intermediate state x' is the same in the forward direction, starting from x, proposing and rejecting x', proposing and accepting x'', and its inverse. This places a limitation of the algorithm and inhibits its application to the variable-dimension simulation, such as RJMCMC samplers.

Green and Mira [1] generalized the delayed rejection algorithm and extended the state space to suit RJMCMC chains. They relaxed the condition on x' and introduced a virtual state x'^*, which brings in more flexibility in the dimensions of variable. We will follow the method and notations in [1] to explain the delayed rejection algorithm applicable for both the fixed dimension and variable dimension cases.

2.3.1 DRA Algorithm

With reference to Figure 2.6, starting from state x, the first proposal is generated from the deterministic function $x' = h_1^+(x, u_1)$ [1], and rejected with probability $1 - \alpha_1(x, x')$,

[1]The subscripts $+$ and $-$ represent the increasing and decreasing of the dimension in state space.

where the acceptance probability α_1 is calculated in the usual way:

$$\alpha_1(x, x') = \min\{1, \frac{\pi(x')g_1'(u_1')}{\pi(x)g_1(u_1)}|\frac{\partial(x', u_1')}{\partial(x, u_1)}|\} \tag{2.39}$$

Then we propose the second candidate $x'' = h_2^+(x, u_1, u_2)$. The dependence on random numbers u_1 and u_2 allows the new value to be influenced by the previously rejected proposal in the first stage. We need to find the suitable acceptance probability $\alpha_2(x, x'')$ for the second stage. This can be derived from the detailed balance at the second stage, which ensures the reversibility of the Markov chain and hence the invariant distribution.

Let $\widehat{\pi}(A, B, C)$ be the probability in equilibrium of starting at state A, proposing and rejecting the first candidate B, and then proposing and accepting the second candidate C, for Borel sets $A, B, C \in \mathcal{X}$. The detailed balance in the second stage requires:

$$\widehat{\pi}(A, \mathcal{X}, C) = \widehat{\pi}(C, \mathcal{X}, A), \text{ for all } A, C \tag{2.40}$$

The left-hand side can be expressed as:

$$\widehat{\pi}(A, B, C) = \int_{(x,x',x'')\in A\times B\times C} dxdu_1du_2\pi(x)g_1(u_1)g_2(u_2)[1 - \alpha_1(x, x')]\alpha_2(x, x'') \tag{2.41}$$

and the right-hand side is:

$$\widehat{\pi}(C, B^*, A) = \int_{(x,x'^*,x'')\in C\times B^*\times A} dx''d\widetilde{u}_1d\widetilde{u}_2\pi(x'')\widetilde{g}_1(\widetilde{u}_1)\widetilde{g}_2(\widetilde{u}_2)[1-\alpha_1(x'', x'^*)]\alpha_2(x'', x) \tag{2.42}$$

where $x'^* = h_1^-(x'', \widetilde{u}_1)$ and $x = h_2^-(x'', \widetilde{u}_1, \widetilde{u}_2)$. In Tierney and Mira [42], B and B^* are chosen to be the same to ensure the integral equality. However, the mapping from (x, u_1, u_2) to $(x'', \widetilde{u}_1, \widetilde{u}_2)$ establishes a diffeomorphism transformation, and the integrands are still equal after changing the variable from x to x'^*. That is to say, the rejected states are not necessarily the same in both directions. The introduced the virtual state x'^* extends the delayed rejection algorithm in [42] to the variable-dimension problems in RJMCMC.

Apply the transformation with x, x', x'' and x'^*, and the change of variable theorem, Equation (2.42) becomes:

$$\widehat{\pi}(C, B^*, A) = \int_{(x'', x'^*, x) \in C \times B^* \times A} dx du_1 du_2 \\ \pi(x'')\widetilde{g_1}(\widetilde{u_1})\widetilde{g_2}(\widetilde{u_2})[1 - \alpha_1(x'', x'^*)]\alpha_2(x'', x)|\frac{\partial(x'', \widetilde{u_1}, \widetilde{u_2})}{\partial(x, u_1, u_2)}| \tag{2.43}$$

Together with Equation (2.41), the detailed balance is satisfied if

$$\pi(x)g_1(u_1)g_2(u_2)[1 - \alpha_1(x, x')]\alpha_2(x, x'') = \\ \pi(x'')\widetilde{g_1}(\widetilde{u_1})\widetilde{g_2}(\widetilde{u_2})[1 - \alpha_1(x'', x'^*)]\alpha_2(x'', x)|\frac{\partial(x'', \widetilde{u_1}, \widetilde{u_2})}{\partial(x, u_1, u_2)}| \tag{2.44}$$

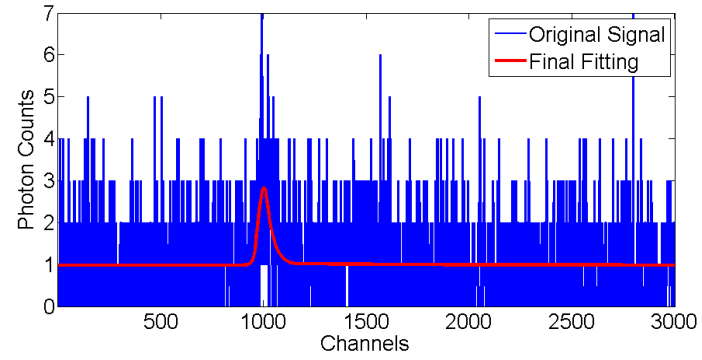
Whereas the required acceptance probability can be solved:

$$\alpha_2(x, x'') = \min\{1, \frac{\pi(x'')}{\pi(x)}\frac{\widetilde{g_1}(\widetilde{u_1})\widetilde{g_2}(\widetilde{u_2})}{g_1(u_1)g_2(u_2)}\frac{[1 - \alpha_1(x'', x'^*)]}{[1 - \alpha_1(x, x'')]}|\frac{\partial(x'', \widetilde{u_1}, \widetilde{u_2})}{\partial(x, u_1, u_2)}|\} \tag{2.45}$$

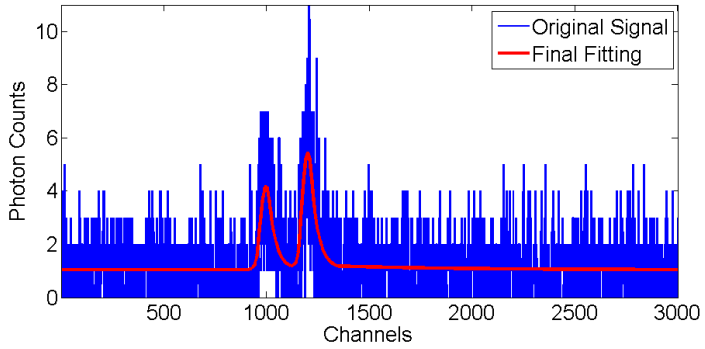
2.3.2 DRA Applied to LaDAR

Slow mixing of the Markov chain indicates a high correlation between the consecutive samples. In general, a fine-tuned proposal distribution can help to improve the mixing performance. However, for LaDAR application, the proposal distribution for t_0 needs to generate both small and large values within the temporal channels. The large step proposals enable the movement between separated channel regions to locate the peak returns, while the small step proposals allow the local exploration of the surrounding areas. To deal with the "conflict" requirements, we employ a delayed rejection step on t_0 to improve the sampling efficiency of the resulting MCMC sampler.

The target of this experiment is to evaluate the effectiveness of the delayed rejection step for t_0 updates in the MCMC estimators. Figure 2.7 shows two simulated histograms: one



(a)



(b)

Figure 2.7: (a) Simulated histogram with a single peak return. (b) Simulated histogram with two separate peak returns.

with a single return placed at an arbitrary location (bin number 1000) and an amplitude of 2 photon counts against a background of 1 photon count; another one with two separate returns located at the 1000 and 1200 bin indices, with the amplitude values of 3 and 4 respectively. The previously unspecified constants were set to $\sigma_\beta = \sigma_B = 0.3$, $a = c = 1.03$, $b = 10,000$ and $d = 1000$. The proposal scales in the delayed rejection step were $\sigma_{t_0}^{\text{step}_1} = 1000$ and $\sigma_{t_0}^{\text{step}_2} = 10$. The allowed maximum number of iterations was 20,000.

Figure 2.8 displays the trace plots for different parameters when using a standard MCMC without a delayed rejection step (scale parameter σ_{t_0} equal to 10 bins). It is observed that the Markov chains in the first column have not escaped from the burn-in periods after 20,000 iterations, which suggests a longer run is required to explore a wider range

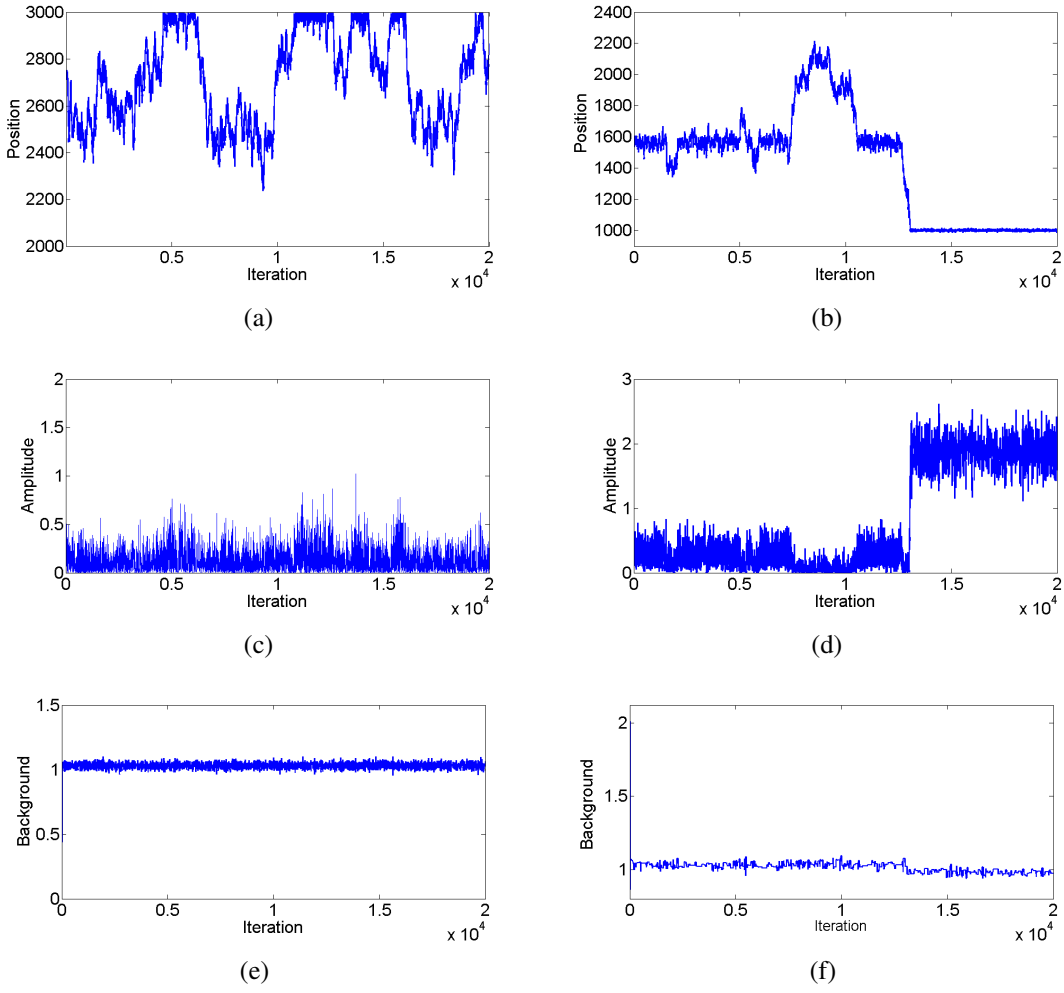
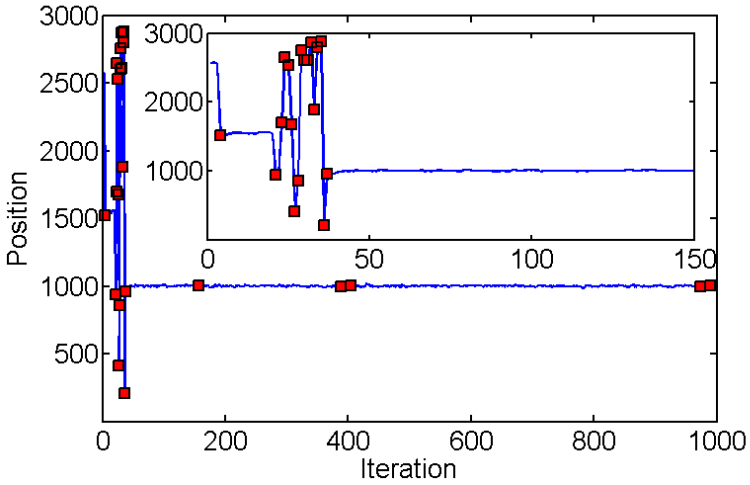


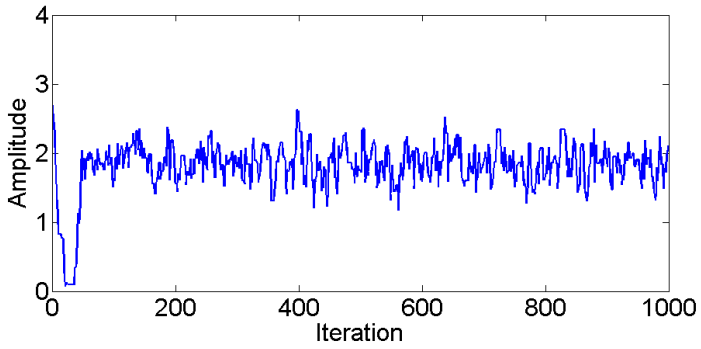
Figure 2.8: Trace plots of different parameters in Figure 2.7(a) when using a standard MCMC algorithm without a delayed rejection step. Figures (a), (c), (e) and Figures (b), (d), (f) are from two independent trials.

of the state space and find the proper region of the peak return. Similar results for the two peaks histogram are presented in the first column of Figure 2.10, where only one of the two returns can be successfully located. These demonstrate that a relatively small value of scale parameter in the random work proposal (Gaussian proposal) will lead to a convergence problem of the Markov chain.

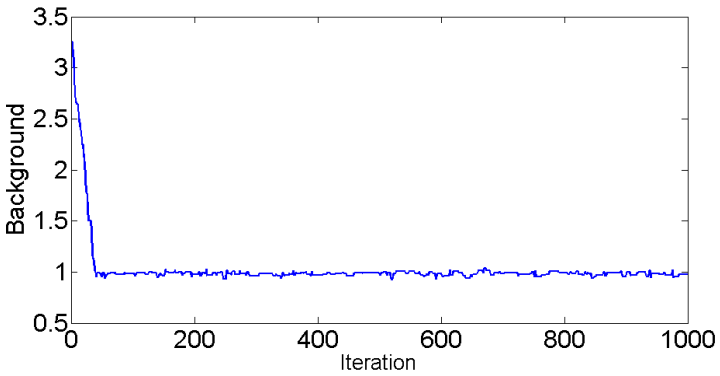
The second column of Figure 2.8 presents Markov chains that have travelled through the



(a)



(b)



(c)

Figure 2.9: Trace plots of different parameters in Figure 2.7(a) when using a standard MCMC algorithm with a delayed rejection step. The accepted first steps are marked with red squares.

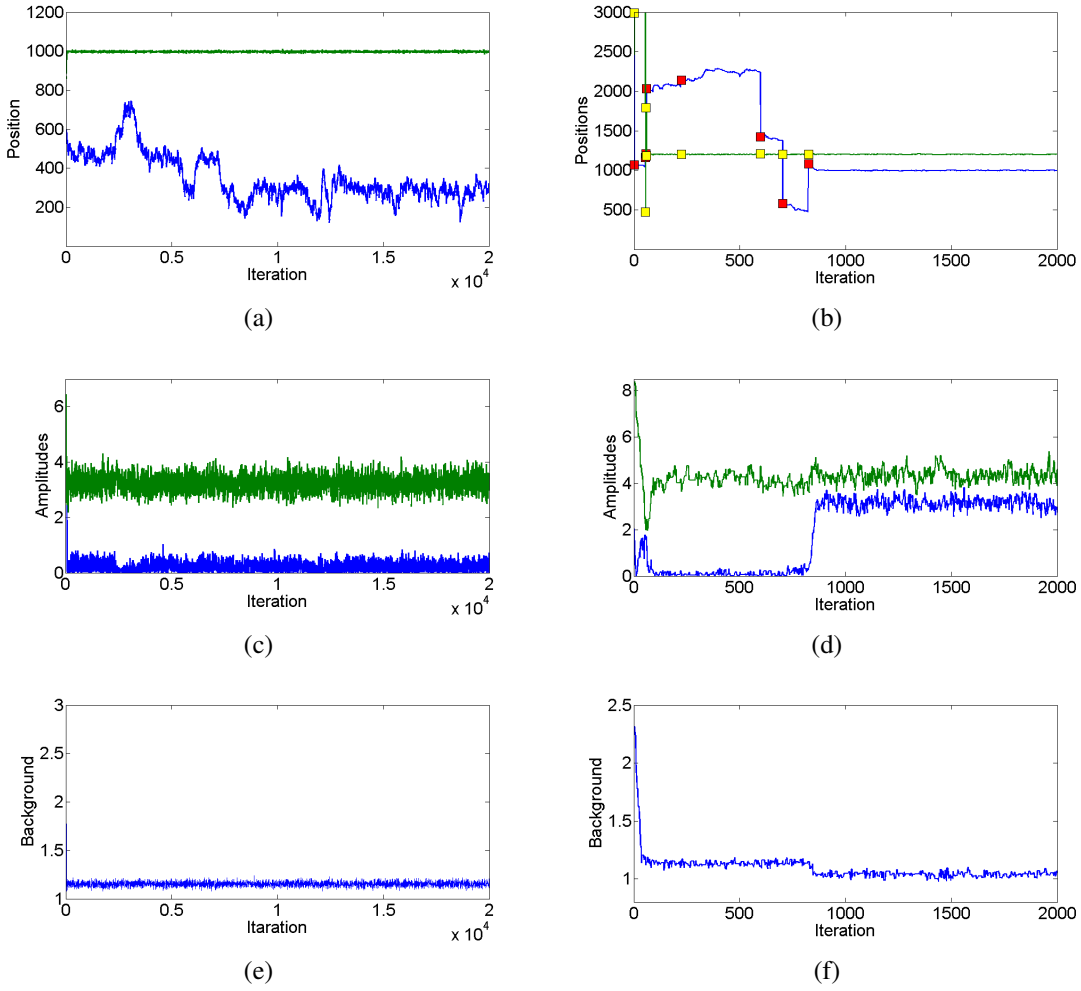


Figure 2.10: Trace plots of different parameters in Figure 2.7(b) when using a standard MCMC algorithm without a delayed rejection step ((a), (c) and (e)), and with a delayed rejection step ((b), (d) and (e)). The accepted first steps are marked with squares.

burn-in periods and reached the covariance stationary status. However, the burn-in periods are considerably long, about 12,000 samples. In addition, from the dynamic variation of the time-series trace plots, we notice that although the parameters are independent in the prior distribution, they have strong dependencies in the posteriors, i.e. the peak amplitude converges almost simultaneously with the peak position. Furthermore, before t_0 and β achieving the convergence, the background value is slightly lower than the true value, since it has been trying to compensate the effects of the false return by increasing Poisson

intensity and so the likelihood value.

Figure 2.9 and the second column of Figure 2.10 display the trace plots of parameters after applying the delayed rejection step. As we observed, initially the first stages with large proposal scale help to explore the temporal channels more quickly which shortens the burn-in period. We also notice that after the burn-in period, first stages can still be accepted, but not quite often; and the accepted ones have very small updating step. Instead, the second steps with smaller scale contribute to the fast refinement of the parameters, and provide the sufficient exploration in the surrounding regions of the found peaks. This verifies the different requirement of proposal scales in the burn-in period and the equilibrium states, which implies the necessity of the delayed rejection step in the MCMC sampler.

2.4 Convergence Assessment

As the chain length goes to infinity, a properly designed MCMC/RJMCMC sampler should theoretically construct a convergent sequence, whose limiting distribution is the true joint posterior distribution of interest. However in practical applications, only a finite number of samples could be produced, which raises the knotty problem of convergence assessment to terminate the Markov process.

Three separate but related issues need to be considered when conducting the convergence assessment [44, 45]: First, evaluate the length of the burn-in period, which is to determine from which observation point the chain has "forgotten" its starting value and "escaped" from its influence. At this point, the chain has reached the stationary distribution and the previous samples should be discarded to eliminate the estimation bias introduced by the transient period. Second, determine if the chain is long enough to fully represent the underlying distribution and achieve its convergence to an asymptotic distribution. Third, evaluate if the samples are adequate to achieve a certain precision of estimation.

Over the last two decades, a number of different convergence diagnostics have been proposed, which can be classified into two categories: the theoretical approaches and empirical approaches. For the theoretical approaches, the attempt is to predetermine the number of iterations required to ensure convergence by analyzing the Markov transition kernel and stationary distribution; a collection of approaches can be found in [46, 47] and references therein. Although they hold formal guarantees, these algorithms are not feasible in practice due to sophisticated mathematical calculation and loose convergence bounds. Therefore, as pointed out in [46], empirical methods are almost always applied, relying on the outputs of the samplers and diagnostics computed from the produced sequence to check convergence.

In the literature, nearly all of the empirical methods seek to diagnose convergence through bias or/and variance evaluation. On the one hand, estimation bias arising from a produced sample can be uncovered by comparing the experimental results with the values of certain statistics that could reasonably reflect the underlying distribution. Methods based on statistical hypothesis tests are prime examples. On the other hand, to provide confidence that a certain level of accuracy has been achieved, some methods, including those built on the concept of confidence region, are intended to indicate the number of samples required for a desired estimation variance.

While these approaches provide evidence of convergence, all the diagnostics are unreliable since in practice the target limiting distribution always remains unknown and it is impossible to conclude with certainty that the finite MCMC/RJMCMC samples are sufficient to cover the whole support of the underlying stationary distribution. From this point of view, we should be cautious about the diagnostic results.

We will present various empirical approaches in this section with brief introductions to their theoretical basis and corresponding implementation procedures, followed by discussions and comments. Our target is to explore the strategies for the *on-the-fly* convergence

monitoring requirement for MCMC and RJMCMC methodology for LaDAR data analysis.

2.4.1 MCMC Convergence Assessment

As shown in Table 2.2, the empirical methods for MCMC convergence assessment can be classified according to different criteria, for example, whether they employ a single chain or multiple chains, whether they are designed for univariate or full joint (multivariate) distributions, whether their results are quantitative or graphical.

Gelman and Rubin (1992)

The convergence diagnostic presented in [48, 49] compares the samples drawn from several independent sequences with different starting points and quantitatively evaluates the mixing by analyzing the within-sequence and between-sequence variance. The idea is that as the number of samples increases, each individual chain will explore larger parts of the parameter space, and consequently, the overall variance and within-sequence variance will both converge to the true model variance. Assume that we simulate $I > 2$ independent sequences initialized with over dispersed starting points, each of length $2T$ and discard the first T samples treated as the burn-in period. For any scalar function $x(\theta)$, we label the t^{th} observation in chain i as x_i^t and calculate the between-sequence variance B:

$$B = \frac{T}{I-1}\sum_{i=1}^{I}(\overline{x_i} - \overline{x_\cdot})^2 \tag{2.46}$$

where

$$\overline{x_i} = \frac{1}{T}\sum_{t=T+1}^{2T} x_i^t, \text{ and } \overline{x_\cdot} = \frac{1}{I}\sum_{i=1}^{I}\overline{x_i} \tag{2.47}$$

Method	Theoretical basis	U/M distribution	S/M chains	Burn-in detection	Convergence diagnostic	x, $\overline{x}$, q	Mixing	Accuracy Evaluation	Ease of Use
Gelman and Rubin	Variance	U	M	No	Yes	x	No	No	a
Raftery and Lewis	2-state Markov Chain	U	S	No	No	q	Yes	Yes	a
Geweke	Compare early/later mean	U	S	No	Yes	$\overline{x}$	Yes	Yes	a
Schruben, Singh and Tierney	Brownian bridge theory	U	S	No	Yes	x	No	No	a
Heidelberger and Welch	Procedure design and half-width test	U	S	Yes	Yes	x	No	Yes	a
Yu and Mykland; Brooks	CUSUM path plots	U	S	No	Yes	x	Yes	No	a
Subsampling	Subsampling, confidence region	M	S	Yes	Yes	x, q	No	Yes	b
Zeller and Min	Conditional posterior densities	M	S	No	Yes	x	No	No	c
Riemann Sums	Riemann Sums	U	S	No	Yes	x	No	No	c

Table 2.2: Summary of convergence assessment methods. U/M distribution: univariate/multivariate distribution. S/M chains: single/multiple chains. x, $\overline{x}$, q: parameter, mean, quantile. "a" is the easiest to use.

The within-sequence variance W is estimated by:

$$W = \frac{1}{I}\sum_{i=1}^{I} s_i^2 \tag{2.48}$$

where

$$s_i^2 = \frac{1}{T-1}\sum_{t=T+1}^{2T}(x_i^t - \overline{x_i})^2 \tag{2.49}$$

The variance of x in the target distribution, V is estimated by:

$$\widehat{V} = \frac{T-1}{T}W + (1+\frac{1}{I})\frac{B}{T} \tag{2.50}$$

The convergence of the Markov chain is monitored by the estimated *potential scale reduction factor* (PSRF),

$$\sqrt{\widehat{R}} = \sqrt{\frac{\widehat{V}}{W}} \tag{2.51}$$

As $T \to \infty$, the total variance estimation $\widehat{V}$ should decrease while the within-sequence variance W might increase, and finally PSRF should theoretically decline to 1. If $\widehat{R}$ is large, it indicates the posterior distribution should be further explored. Once PSRF close to 1, we assume the Markov chain convergences to the target distribution.

Although variance measurement based on multiple chains could detect whether the underlying stationary distribution has been fully explored and whether the chains have converged to the same limiting distribution, there are still a number of criticisms. First of all, finding the over dispersed starting points with respect to the target distribution is not an easy task. In addition, the approach is only capable of dealing with a univariate distribution. Moreover, this diagnostic cannot detect the burn-in period, and discarding the large enough samples from early iterations in all the sequences exhibits inefficiency for chain generation.

Raftery and Lewis (1992)

The Raftery and Lewis [50] method focuses on the estimation accuracy of the quantile q of the posterior distribution of a function U of parameter $\mathbf{x}$, expressed as $P(U(\mathbf{x}) \leq u|\text{Data}) = q$. It is intended to predict in advance the length of the burn-in period M and the number of samples N in a stationary distribution for a desired precision level and a thinning factor k, which is to extract every k^{th} iteration for final analysis regarding the sample dependency as well as the storage consumption. The chain is terminated when the estimated quantile lies within $\pm r$ (e.g. ± 0.005) of the true value with probability s (e.g. 0.95).

The approach is based on two-state Markov chain theory and the central limit theorem. The original samples are mapped into a binary $0-1$ sequence in which 1 indicates $U(\mathbf{x}_t)$ is less than a predefined cut-off value u. Following this, a subsequence Z_t^k with thinning parameter k is constructed to approximate a Markov process by comparing the Bayesian information criterion (BIC) between the first and the second order Markov models. The burn-in period is then deduced by evaluating how closely the multi-step transition matrix of Z_t^k approaches its stationary distribution. The sample size corresponding to the precision constraint is finally obtained using a normal approximation for the distribution of the sample mean in Z_t^k.

The diagnostic process is implemented in the Fortran program `gibbsit` (Fortran functions written by Lewis) and the CODA package using the S-plus language. There are a few steps to follow in order to obtain the value of M, N and k.

1. Calculate the minimum size of a pilot run using the following expression. This is used as an input of the program to predict M, N and k for a chosen quantile q to obtain the specific precision.

$$N_{\text{min}} = \Phi^{-1}(\frac{1}{2}(s+1))^2 q(1-q)/r^2 \tag{2.52}$$

where $\Phi(\cdot)$ is the standard normal cumulative distribution function.

2. Call `gibbsit` or the corresponding function in the CODA package. The returned values are the estimate of M, N and k.

3. It is recommended to call the function again after $(M + N)$ iterations to verify the chain length predicted based on the pilot run is sufficiently long. If $(M+N)$ values from the second run are appreciably more than that from the first call, then further iterations are required.

As pointed out in [50], it is better to go through the procedure for each of the quantiles of interest and chose the largest $(M+N)$. Due to the difficulty in tail quantile estimates, we can choose $q = 0.025$ and $q = 0.975$ as a reasonable routine practice.

While this algorithm could help to predict the chain length, it is unable to detect that convergence is achieved when the chain reaches $(M+N)$ iterations. However, the output of the program can be used to evaluate the Markov chain performance. A long transition period M implies slow convergence to the stationary distribution, while a large ratio of $(M + N)/N_{\text{min}}$ or/and k larger than one indicate a strong autocorrelation, which might suggests we should choose a better proposal distribution or a more reasonable starting value. Another limitation is that it is specifically designed for the particular quantiles and it is necessary to re-diagnose for each of them. Nevertheless, this does not provide information about the convergence of the entire chain.

Geweke (1992)

The objective of [51] is to estimate the mean of some function of the parameters $g(\theta)$ by comparing two sub-sequences in the early and later stages of the chain. If the chain has converged, estimates based on these two sets of samples should be the same. To measure the differences between the locations, a statistic is constructed which includes asymptotic variance estimated based on the spectrum density. The underlying assumption

is the existence of a spectrum density $S_G(\omega)$ for the function g with no discontinuities at zero frequency.

Consider two subsequences after the burn-in period $\theta_A^t : t = 1, ..., n_A$ and $\theta_B^t : t = n_B, ..., n$, where $1 < n_A < n_B < n$ with suggested values $n_A = 0.1n$ and $n_B = 0.5n$. With respect to the central limit theorem, the distribution of the constructed diagnostic statistic Z_n approaches a standard normal distribution as $n \to \infty$:

$$Z_n = \frac{\overline{\theta_G^A} - \overline{\theta_G^B}}{(\frac{S_G^A(0)}{n_A} + \frac{S_G^B(0)}{n_B})^{1/2}} \overset{n\to(\infty)}{\to} N(0,1) \tag{2.53}$$

where $\overline{\theta_G^A}$ and $\overline{\theta_G^B}$ denotes the sample means while $S_G^A(0)$ and $S_G^B(0)$ are the spectral density estimates at $\omega = 0$ for two sub-chains respectively.

Geweke defines the square root of asymptotic variance $S_G(0)/n$ as the *numerical standard error* (NSE) to assess numerical accuracy. Another parameter introduced is *relative numerical efficiency* (RNE), which is var$[g(\theta)]/S_G(0)$. It provides information about the sample size required for a given precision degree.

The merits of this approach lies in that it attempts to use a single chain to address both of the bias and variance problems. Also, according to [46], it might be extended to detect the convergence of joint posterior density. However, the drawbacks should not be neglected: the asymptotic variance is sensitive to the spectral window and also there is no explicit procedure for its application and hence it rests on the experience of statisticians.

Schruben, Singh and Tierney (1983)

The stationary assessment method of [52] is founded on the statistical hypothesis test, where the null hypothesis is that the samples are drawn from a covariance stationary process which is $\phi-$mixing. After introducing a test statistic based on Brownian bridge theory, we can either reject or accept the hypothesis by testing whether this statistic lies inside or outside a *critical region*, which is determined by the probability density of the

test statistic given that the null hypothesis is true, and the level of significance.

Suppose θ^j is the j^{th} sample at time j, n is the total number of iterations, $[\cdot]$ is the rounding operator and:

$$T_k = \sum_{j=1}^{k} \theta^j; \text{with } T_0 = 0 \tag{2.54}$$

$$\overline{\theta} = \frac{\sum_{j=1}^{n} \theta^j}{n} \tag{2.55}$$

A sequence $B_n(t)$ is built up using T_k from the subsets of θ^j:

$$B_n(t) = \frac{T_{[nt]} - [nt]\overline{\theta}}{\sqrt{nS(0)}}, 0 \leq t \leq 1 \tag{2.56}$$

where $S(0)$ is an estimate of the spectral density at zero frequency, and coordinates t are on the value $t = \frac{1}{n}, \frac{2}{n}, ..., 1$. For large n, $B_n = B_n(t), 0 \leq t \leq 1$ converges in distribution to a Brownian bridge. The test statistic CVM(B_n) constructed using Bn follows a standard Cramer-von Mises distribution as $n \to \infty$:

$$\text{CVM}(B_n) = \int_0^1 B_n(t)^2 dt \tag{2.57}$$

and the integral can be numerically approximated using Simpson's rule. The stationary test is one-sided and rejection occurs when the $p-$value is greater than $1 - \alpha$ where α is a critical level (the default value is 0.05).

Although this approach can detect the convergence, the initial transient issue remains unsolved. The extended method proposed by Heidelberger and Welch intends to address this problem and monitor when the Markov chain converges to its underlying stationary distribution.

Heidelberger and Welch (1983)

As shown in Figure 2.11, the diagnostic procedure contains two parts: first, the *station-*

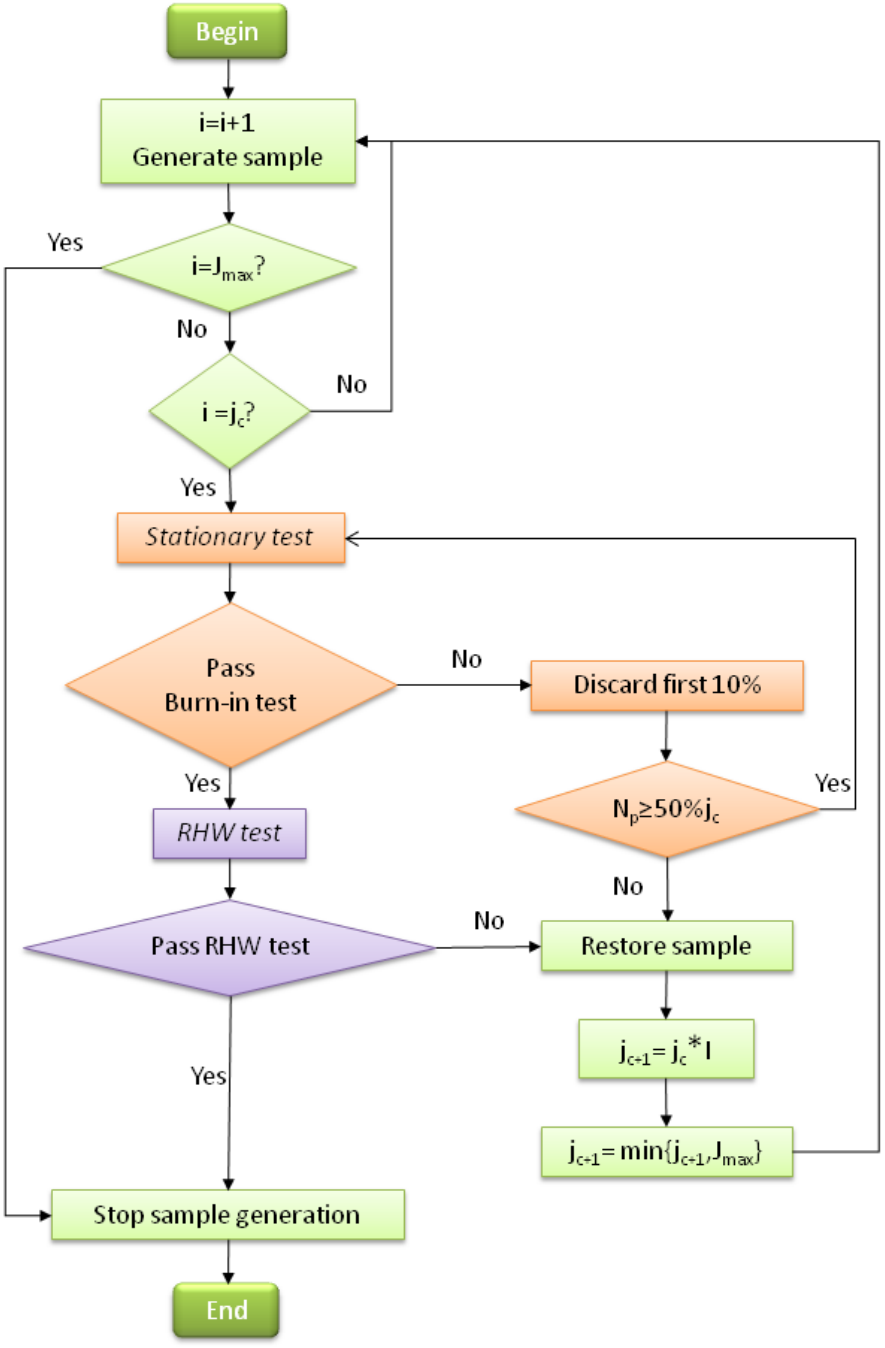


Figure 2.11: Heidelberger and Welch convergence diagnostic.

arity test with the aim of initial transient (burn-in period) removal, where a Cramer-von Mises statistic is defined to test the null hypothesis that the sequence is from a covariance stationarity process with no transient period; second, the *relative half-width (RHW) test*, which evaluates whether the Markov chain has produced sufficient samples to satisfy a desired accuracy requirement for the mean estimates. This procedure can be applied to a single chain from any Markov chain Monte Carlo algorithm regardless of discrete or continuous event simulation as stated in [46].

The stationarity test is applied to the entire sequence. If the null hypothesis is rejected, the initial 10% of the original sequence is discarded as a burn-in period and the test is repeated. This is iterated until the stationarity test is passed and the proportion of the remaining samples is no less than 50%, otherwise the stationarity test fails. In the former case, we calculate the RHW of the $(1-\alpha)$ level confidence interval of the mean estimate $\widehat{\theta}$ (here, either k or ϕ_k), which is defined as

$$\text{RHW} = \frac{z_{1-\alpha/2}\sqrt{\widehat{S}(0)/n_p}}{\widehat{\theta}} \tag{2.58}$$

where $z_{1-\alpha/2}$ is the z-score of the $100(1-\alpha/2)^{\text{th}}$ percentile, $\widehat{S}(0)/n_p$ is the asymptotic variance of the mean estimate for the truncated sequence of length n_p, and $\widehat{S}(0)$ is the power spectrum density at zero frequency. If RHW is less than a predefined threshold, we conclude convergence and terminate the chain generation. If either the stationarity test or the RHW test fails, we restore the removed samples and increase the run length until both succeed, or the chain length reaches an allowed maximum limit.

With the transient removal procedure, we can reduce the estimation bias and narrow down the confidence intervals with the same or even a shorter chain length. However, J_{max} should be chosen with caution to ensure it is longer than the length of burn-in period.

Yu and Mykland (1998); Brooks (1998)

A graphical method proposed by [53] aims to assess the convergence by monitoring the chain mixing performance using CUSUM path plots. It is applied to a univariate scalar summary function $\theta(\mathbf{x})$ from a single chain. The burn-in period n_0 is first discarded before constructing the CUSUM plots obeying the following steps:

1. Calculate the mean of the summary statistic: $\widehat{\mu} = \frac{1}{n-n_0} \sum_{j=n_0+1}^{n} \theta(\mathbf{x})^{(j)}$
2. Calculate the observed CUSUM or partial sum $\widehat{S}_t = \sum_{j=n_0+1}^{t} [\theta(\mathbf{x}^{(j)}) - \widehat{\mu}]$
3. Plot $\widehat{S}_t$ for $t = n_0 + 1, ..., n$.

As stated by Yu and Mykland, the chain has good mixing properties if the CUSUM plots are irregular and centred around 0. In contrast, if the graph is smooth and apart from 0, mixing is slow.

This approach provides an intuitive indication of mixing performance. It is simple and convenient regarding implementation and interpretation. However, it also holds distinct disadvantages. First, only a univariate summary function can be monitored rather than the full joint posterior distribution. Second, the length of burn-in period must be determined with the help of other diagnostic methods. Third, a good mixing property is not a sufficient and necessary condition for convergence to a stationary distribution. Last but not least, as a graphical scheme, it is not an ideal choice for numerical analysis.

Brooks [54] updated the idea of CUSUM in [53] to a quantitative method by introducing a test statistic and performing the statistical hypothesis test. Let:

$$d_t = \begin{cases} 1 & \text{if } S_{t-1} > S_t \text{ and } S_t < S_{t+1}, \text{ or } S_{t-1} < S_t \text{ and } S_t > S_{t+1} \\ 0 & \text{otherwise} \end{cases} \tag{2.59}$$

$$D_{n_0,n} = \frac{1}{n = n_0} \sum_{t=n_0+1}^{n-1} d_t \tag{2.60}$$

where the arithmetical mean $D_{n_0,n}$ is the constructed test statistic which follows the binomial distribution. If $n - n_0$ is large, $D_{n_0,n}$ approximates a normal distribution. This is a two-sided test and we can determine the convergence by testing whether $D_{n+0,n}$ lies inside the critical region.

Subsampling

The method investigated by [45] is based on the basic ideas of subsampling, which is to approximate the sampling distribution of a statistic on a whole chain by computing the values of the same statistic over numerous smaller subsets of the data with the retained dependence structure of the entire sequence. It rests on two assumptions for a multivariate time series X_s: *asymptotic* independence and *asymptotic* stationarity with consideration of the effect arising from limited sample size. The convergence assessment is carried out by evaluating the properties of the confidence region in the empirical distribution.

The subsampling procedure is progressed as follows: suppose T_N is a statistic of interest, which is a function of data sequence $\{X_1, X_2, ..., X_N\}$. It performs as an estimator of an unknown parameter θ. Assume T_N is consistent for θ as $N \to \infty$ and according the large number law, T_N will be suitably centred and normalized. Let

$$J_N(x) = \Pr[\tau_N \|T_N - \theta\| \leq x] \tag{2.61}$$

be the quantile over N samples where $\|\cdot\|$ is a norm and $\{\tau_n, n = 1, 2, ...\}$ is an increasing sequence satisfying:

$$\{\tau_n, n = 1, 2, ...\} \overset{n\to\infty}{\to} \infty \tag{2.62}$$

Now suppose $J(\cdot)$ is a nondegenerate continuous distribution function such that:

$$J_N(x) \overset{N\to\infty}{\to} J(x) \text{ for all } x \tag{2.63}$$

Since practically θ is always beyond our knowledge and only one T_N value is available from one sequence realization, it is impossible to estimate the quantile $J_N(x)$. Therefore, we consider to use B subsample values $\{T_{i,b}, i = 1, ..., B\}$, each of which is calculated from b consecutive observations $(X_i, X_{i+1}, ..., X_{i+b-1})$, to approximate an empirical distribution $L_N(x)$, which will be employed as an estimator of the limit distribution $J(\cdot)$:

$$L_N(x) = \frac{1}{B} 1\{\tau_n \|T_{i,b} - T_n\| \leq x\} \tag{2.64}$$

$$L_N(x) \overset{N\to\infty}{\to} J(\cdot) \tag{2.65}$$

Now we can construct a confidence region for θ using the quantile of L_n with asymptotic converge probability equal to the nominal $(1 - \alpha)$:

$$\{\theta : \tau_n \|T_n - \theta\| \leq L_n^{-1}(1 - \alpha)\} \tag{2.66}$$

Generally, $\tau_n = \sqrt{n}$ and the block size b is proportional to n^γ with $\gamma \in (0, 1)$. Realizing that the range of a $(1 - \alpha)\%$ confidence region is proportional to $1/\sqrt{n}$ no matter that T_N is chosen to be the mean or the quantile, the convergence diagnostic can be formulated as follows:

1. Estimation of burn-in period.

   Choose T_N to be the q quantile (e.g. $q = 0.90$). Construct the $(1 - \alpha)\%$ (e.g. $\alpha = 0.05$) confidence region for different increasing value of n and plot it versus $1/\sqrt{n}$. The burn-in time n_0 is the point after which the plot is linear for $n > n_0$. Linearity can be tested either graphically or quantitatively as explained in [45].

2. Estimation of chain length.

   Choose T_N to be the mean and construct the $(1 - \alpha)\%$ confidence region for increasing n. Stop the simulation when the range of confidence region is smaller than

a prespecified measure of accuracy.

This diagnostic holds several significant benefits. First, noticing the fact that the MCMC output is not exactly stationary, this method is built upon asymptotic stationarity. Second, by choosing the supreme norm in Equation (2.64), it can be applied for multivariate distributions. Third, both the transient period and the chain length for a particular precision level can be estimated using a single MCMC chain either graphically or quantitatively.

Zeller and Min

Zeller and Min [55] not only aim to determine the convergence in a distribution but also determine whether the MCMC outputs converge to a correct distribution. This is realized by comparing the posterior probability estimated from the generated samples for a specific parameter set with its theoretical value, or checking if the two estimated posterior densities are equal or at least close to each other. The approach relies on the assumption that the whole parameter set θ is *bimodal*, that is it can be factorized as $\theta = (\alpha, \beta)$; and the conditional posterior density $p(\alpha|\beta, D)$ and $p(\beta|\alpha, D)$ can be derived explicitly, where D stands for the data y and prior information is I_0.

Suppose that (α_i, β_i) represents the selected parameter point, then the marginal posterior density can be approximated by:

$$\widehat{p}_N(\alpha_i|D) = \frac{1}{N}\sum_{j=1}^{N} p(\alpha_i|\beta^{(j)}, D) \tag{2.67}$$

$$\widehat{p}_N(\beta_i|D) = \frac{1}{N}\sum_{j=1}^{N} p(\beta_i|\alpha^{(j)}, D) \tag{2.68}$$

where $(\alpha^{(j)}, \beta^{(j)}), j = 1, 2, ..., N$ denotes the MCMC samples drawn from the conditional densities $p(\alpha|\beta, D)$ and $p(\beta|\alpha, D)$.

Three convergence criteria are derived as follows:

1. The anchored ratio convergence criterion (ARC^2):

   Define an anchored ratio θ_{12} for two points (α_1, β_1) and (α_2, β_2):

$$\theta_{12} = \frac{p(\alpha_1, \beta_1|D)}{p(\alpha_2, \beta_2|D)} = \frac{\pi(\alpha_1, \beta_1|I_0)l(y|\alpha_1, \beta_1)}{\pi(\alpha_2, \beta_2|I_0)l(y|\alpha_2, \beta_2)} \tag{2.69}$$

   From Equation (2.67) and (2.68), calculate:

$$\widehat{\theta}_{12}(N) = \frac{\widehat{p}_N(\alpha_1|D)p(\beta_1|\alpha_1, D)}{\widehat{p}_N(\alpha_2|D)p(\beta_2|\alpha_2, D)} \tag{2.70}$$

$$\widetilde{\theta}_{12}(N) = \frac{\widehat{p}_N(\beta_1|D)p(\alpha_1|\beta_1, D)}{\widehat{p}_N(\beta_2|D)p(\alpha_2|\beta_2, D)} \tag{2.71}$$

   The MCMC outputs should have converged if $\widehat{\theta}_{12}(N) \approx \widetilde{\theta}_{12}(N) \approx \theta_{12}$.

2. The difference convergence criterion (DC^2):

   Compute $\widehat{\eta}_i(N)$ as follows:

$$\widehat{\eta}_i(N) = \widehat{p}_N(\alpha_i|D)p(\beta_i|\alpha_i, D) - \widehat{p}_N(\beta_i|D)p(\alpha_i|\beta_i, D) \tag{2.72}$$

   Based on the relationship:

$$\begin{aligned} p(\alpha, \beta|D) &= p(\alpha|D)p(\beta|\alpha, D) \\ &= p(\beta|D)p(\alpha|\beta, D) \end{aligned} \tag{2.73}$$

   We can conclude the convergence if $\widehat{\eta}_i(N) \approx 0$.

3. The ratio convergence criterion (RC^2):

   Define the convergence diagnostic $\widehat{\gamma}_i(N)$ based on (2.73) as:

$$\widehat{\gamma}_i(N) = \frac{\widehat{p}_N(\alpha_i|D)p(\beta_i|\alpha_i, D)}{\widehat{p}_N(\beta_i|D)p(\alpha_i|\beta_i, D)} \tag{2.74}$$

   This ratio should satisfy $\widehat{\gamma}_i(N) \approx 1$ if the sampler has converged.

These convergence diagnostics are intended to quantitatively detect estimation bias using a single chain. However, a number of criticisms have been made. It is problem-dependent and hence so is the coding. The parameter factorization and conditional posterior densities derivation requirements form the application limitations of this method.

Riemann Sums (2001)

In [56], convergence diagnostics are in the form of Riemann sums, which are derived to approximate the Monte Carlo integral that is expressed as:

$$E[h(x)] = \int_{\Re} h(x)f(x)dx \tag{2.75}$$

where $h(x)$ is a function of interest. The Riemann sum is defined as:

$$\delta_T^h = \sum_{i=1}^{T-1} (x^{[i+1]} - x^{[i]})h(x^{[i]})f(x^{[i]}) \tag{2.76}$$

where $x^{[1]} \leq ... \leq x^{[T]}$ is the ordered sample drawn from density f or a proposal density after burn-in period. If f is a univariate distribution or a univariate marginal density in a closed form, $h(x)$ can be set to 1. As the chain converges towards its stationary distribution, δ_T^h should converge to 1.

In the multivariate case where f can not be written in the closed form, the marginal density of the l^{th} component is approximated using Rao-Blackwellized estimation:

$$T^{-1} \sum_{k=1}^{T} \pi_l(x_l^{[t]} | x_{-l}^{(k)}) \tag{2.77}$$

Where $x_{-l} = (x_1, ..., x_{l-1}, x_{l+1}, ..., x_p)$. The Riemann sum now becomes:

$$\Delta_T^1(l) = T^{-1} \sum_{t=1}^{T-1} (x_l^{[t+1]} - x_l^{[t]})(\sum_{k=1}^{T} \pi_l(x_l^{[t]} | x_{-l}^{(k)})) \tag{2.78}$$

Successful convergence is diagnosed when $\Delta_T^1(l)$ is close to 1.

To use this method, we must know the univariate density f while in the multivariate case, the univariate marginal density must either be written in the closed form or estimated using the analytical full conditional distribution $\pi_l(x_l^{[t]}|x_{-l}^{(k)})$.

2.4.2 RJMCMC Convergence Assessment

Convergence assessment for the RJMCMC becomes much more difficult since the iterations not only update the parameters through within-model moves but also the model dimensions via trans-model jumps. One common way to cope with the convergence assessment difficulty in RJMCMC is to define a model indicator which uniquely identifies different models, for instance, it can be the number of components in the mixture model, such the number of peak returns in the LaDAR histogram. We can first check the convergence for the model indicator, and then monitor the convergence within each individual model after it appears to have reached stationarity [57]. However, some models may be seldom visited even in a long run and hence a convergence diagnostic for these models is almost impossible to estimate. Thus, there might be a problem associated with the selection of a model to monitor.

One way to circumvent this problem is to parameterize the model such that its interpretation does not change as the simulation proceeds even when trans-model jumps occur. Based on this idea, [58] proposed a method based upon a two-way, ANOVA-type decomposition of the simulation to extend the work of [48]. In [59], a single scalar summary function of the parameter set is defined and the variance defined in [48] is now split both between sequences and between models.

Specifically, in [59] six statistics (within/between-model variances, within/between chain variances, within/between model within chain variances) are constructed and three pair-

wise ratios are created respectively. As the RJMCMC chain is much more complex than the MCMC counterpart, all of the original six parameters are monitored to gain insight into the chain mixing rather than just to evaluate how close these three ratios are to 1. The simulation based conclusion is that although the between chain and within chain variances have stabilized, the other four statistics may show dramatic sudden changes in value and differ significantly from one when a chain visits a rather improbable model. This phenomenon indicates that even though some chains have already approached their steady states, they may have not visited some models. From this point of view, more iterations may be required to further enhance the between-model mixing.

However, as discussed in [60] this method is not always feasible since for some of the statistical models, it is not always possible to define such parameters with persistent interpretation. Moreover, we need to be cautious that the convergence monitored from these parameters may not describe the full convergence properties of the chain as a whole.

The method in [60] intends to assess the convergence performance of an RJMCMC sampler using nonparametric techniques based on a statistical hypothesis test. The distance measures between two or more sets of Markov chains are defined, which can quantify the similarity between multiple replications and are then used to evaluate the convergence with the assumption that the distance should be small when the chains have converged to the stationary condition. Two explicit goodness-of-fit tests for homogeneity are examined in detail, the chi-squared test and the two-sample Kolmogorov-Smirnov test. They can be applied to either the RJMCMC or the birth-death approach.

This technique gives emphasis to trans-model convergence as it considers how the observed events are allocated to a set of models (i.e. the missing data) and measures the allocation differences among multiple MCMC outputs by evaluating the distance in the obtained probability distributions. However, it does not provide sufficient information about how closely the within-model parameters have approached their underlying sta-

tionary distribution.

2.4.3 Convergence Diagnostic Selection

The simulation efficiency of LaDAR analysis, either using MCMC for single opaque surface or RJMCMC for multiple distributed surfaces, can be captured by the amount of computations in the sampling proceeding measured by the Markov chain length. We expect a convergence diagnostic which can indicate the mixing performance of the output sequence, detect the burn-in period and determine the proper chain length to achieve a certain accuracy level. Based on the discussions of the diagnostics reviewed in Section 2.4.1 and 2.4.2, we choose two approaches, the Gelman and Rubin diagnostic and the Heidelberger and Welch diagnostic, to satisfy these requirements for LaDAR analysis.

The Gelman and Rubin diagnostic is particularly suitable for multi-modal posterior distribution, since the between sequence comparison provides information about whether the sequences are exploring the local or global regions in the state space, and implies whether the samples have travelled across the valley between peaks in the posterior distribution. In the case of LaDAR, we might observe complex multi-modal posterior distributions. The Gelman and Rubin diagnostic can help to detect the local optima, particularly for RJMCMC sequences stuck in one particular model, and provide reliable evidence for a "global" convergence. However, it cannot determine the length of burn-in period and must rely on multiple chains.

The Heidelberger and Welch diagnostic can detect and remove samples in the transient period using a single chain. Since the designed procedure is applicable to both discrete and continuous simulation events, it is suitable to assess the convergence for both the model indicator, the number of peaks in LaDAR signals, and the related model parameter, i.e. the peak amplitudes, positions and the background level.

2.5 Simulation Comparison for Full Waveform LaDAR Analysis: Cross-correlation, MCMC and RJMCMC

In this section, we present the analysis of images acquired under bright daylight conditions of two distant outdoor scenes, comparing methods based on cross-correlation and fixed and variable dimension Markov chain Monte Carlo analysis. Our images are of a life-sized mannequin (a human figure) in full view of the sensor, and of the same mannequin partially concealed behind a fence. The data were acquired at a range of approximately 325 meters. The equivalent scene dimensions were 0.8m width by 2.0m height, and the scanned image resolution was 32 by 128 pixels for the whole mannequin. The pulse repetition frequency was 2MHz, resulting in an average optical power of 40μW. The pixel dwell time was 1.0 sec.

To assess the ability of the RJMCMC algorithm for multiple peak detection and particularly the resolution capacity for closely separated peaks, we also set up a remote target containing several distributed surfaces with known separations, which provides the ground truth and allows us to compare the performance with cross-correlation method.

2.5.1 Mannequin in Full View: Cross-correlation and MCMC

In the first example, the mannequin is in full view, standing in front of a concrete pillar, as shown in Figure 2.13. It was anticipated that the majority of pixels would have clear and distinct, single returns from the surface of either the mannequin or the pillar. Given the divergence of the beam, there may be some mixed pixels at the occluding boundary of the mannequin, and there may be pixels with no return as they miss the targets all together. In short, this is a situation in which a cross-correlation detector based on the system instrumental response should perform well and there should be questionable need for the

added complexity of Markov chain Monte Carlo analyses. Further, since the expectation in processing this data set is to estimate the range of a single surface return from either the mannequin or the pillar, we apply the fixed dimension MCMC approach to avoid redundant computation caused by trans-dimension jumps.

As stated in Section 2.2.2.2, the unknowns (t_0, β, B) subject to inference have independent priors. The shape parameters a, c equal to 6 and 1.5, and the scale parameters b, d are set to $\frac{\max(\mathbf{y})/2}{6}$ and $\text{mean}(\mathbf{y})$ respectively. The previously unspecified proposal distributions are set as follows: all of the parameter updates employ the Gaussian random walk whose proposal means are the current sample values. The standard deviations for amplitude (σ_β) and background (σ_B) are both 0.3. For position updates, a delayed rejection step [22] is carried out to allow movement between posterior estimates that correspond to more widely separated channels. When using delayed rejection, the scale in each step is characterized by $\sigma_{t_0}^{\text{step}_1} = 1000$ and $\sigma_{t_0}^{\text{step}_2} = 10$ respectively.

We first generate multiple chains for each pixel and evaluate the convergence. After finding a safe convergence length, we then run single MCMC chains with $k = 1$ on all the pixels with a bounded number of iterations (5000) including the 500 samples burn-in period. This is consistent with the initial estimate. Subsequently, to assess the convergence of the MCMC chains, we produce four independent sequences for each pixel, and monitor the Gelman and Rubin diagnostic statistic (PSRF) defined in Section 2.4.1 every 100 samples. The chain generation is terminated when the convergence is concluded, that is when the PSRF reduces to less than a preset threshold 1.002, at which the posterior distributions $p(t_0|\mathbf{y}, k = 1)$ obtained from all the sample trajectories becomes approximately the same.

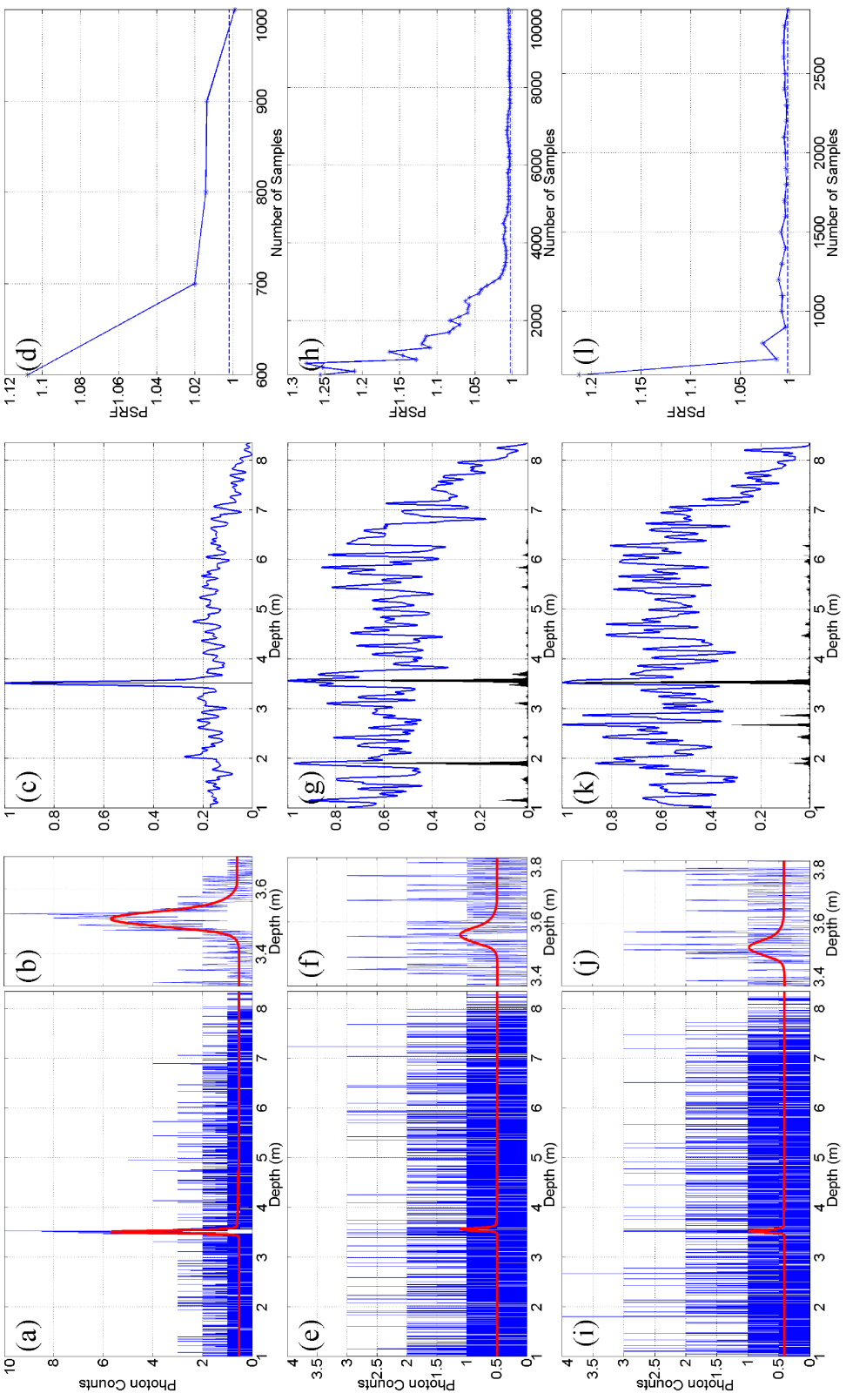


Figure 2.12: Analysis of time-of-flight ladar data, in which the histogram bins have been converted to relative depth in meters. The first column shows the raw pixel data (in blue). The second column magnifies the plots of signal peaks in the first column. The third column shows the normalized cross-correlation values (blue curves) and the frequencies of positions (black bars) obtained from the MCMC samplers. The last column tracks the corresponding PSRF values against the number of samples. The final fit estimations (from MCMC) are the red curves in the first column.

Figure 2.12(a) presents a representative pixel with a single distinct return. For this type of pixel data, there is a clear, sharp peak in the normalised cross correlation plot and a distinct preference in the frequency of positions obtained from MCMC sequences. Their maximum values are both located in the same channel index as shown in Figure 2.12(c). In this circumstance, the cross-correlation approach can easily detect the surface return, and according to Figure 2.12(d), MCMC chains can converge rapidly with a small number of samples (about 500 samples after the burn-in period) due to the simplicity of parameter space.

For the low amplitude return in Figure 2.12(e), the cross-correlation approach gives several extrema as displayed in Figure 2.12(g). Such low amplitude may be caused primarily by lower reflectance back towards the receiver, either because of the material properties or its angle to the beam direction. In this case, it is difficult to decide with certainty where the surface return is located, although we can always define it to be the one corresponding to the maximum cross-correlation value. In comparison, the power of the MCMC methodology lies in supplying Bayesian evidence of the final answer. In other words, the histogram of t_0 indicates the posterior distribution of the estimates. As the parameter space becomes more complex, the posterior distribution is spread over a wider channel range and becomes bi-modal, which in turn results in a slower convergence rate and an increased chain length in excess of 4000 samples.

Another example is shown in Figure 2.12(k). For this pixel, the bin index for the maximum cross-correlation does not equal the one for the $p(t_0|\mathbf{y}, k = 1)$ posterior mode. Hence, the MCMC chain gives a different and better substantiated estimate of the true value, further demonstrating the power of the Bayesian approach.

3D images based on these two methods are provided in Figure 2.13, where a target range gate is set and those pixels with with target position estimates beyond this pre-set gate are treated as zero return. It is observed that there are a few more pixels beyond the target

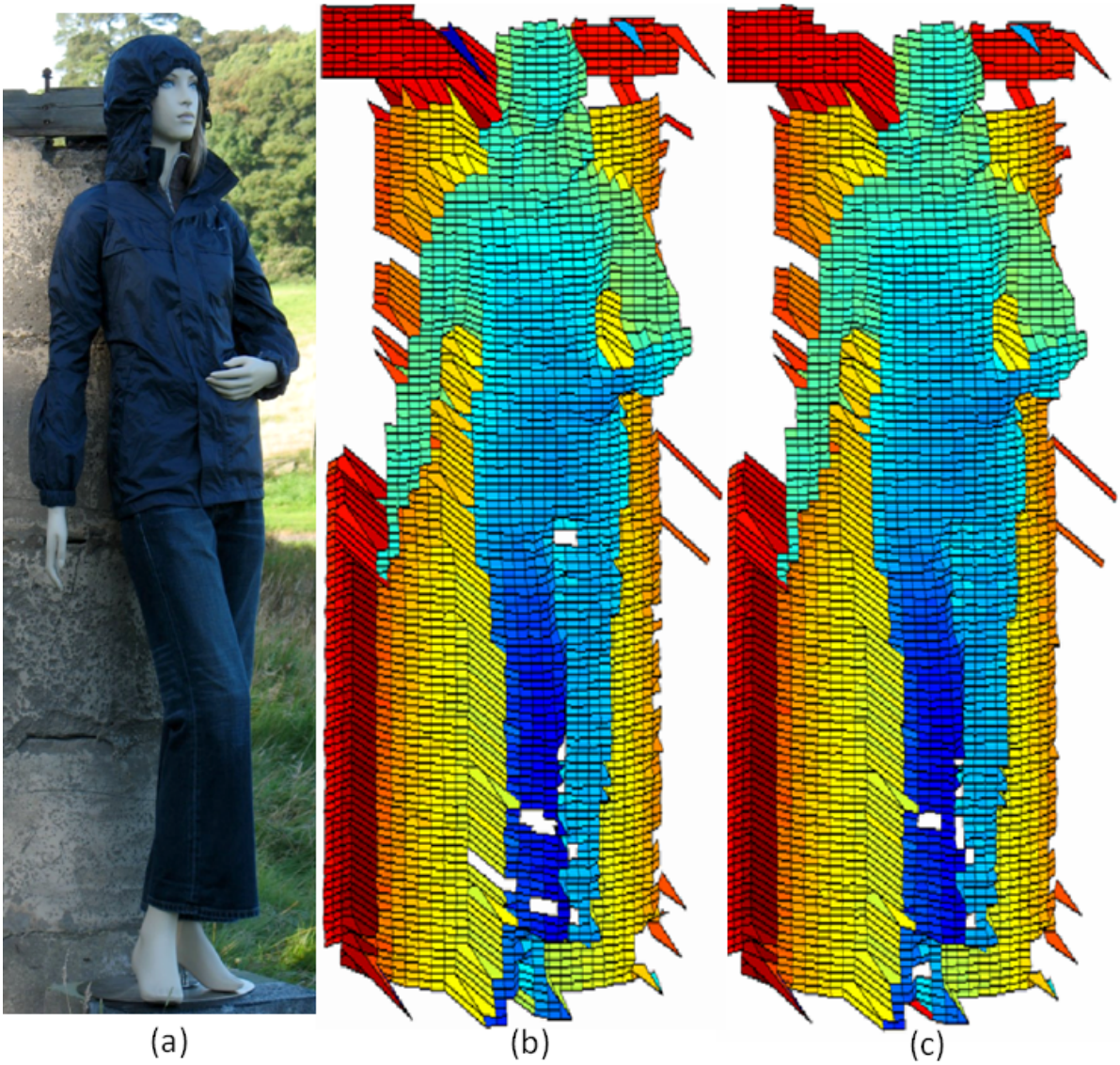


Figure 2.13: 32x128 pixel image of a life-sized mannequin scanned at a distance of 325m in daylight conditions. (a) Photograph of the 1.8m tall mannequin in the scan position. (b) and (c) Three-dimensional plots of the processed depth information using the cross-correlation and MCMC methods, respectively. Empty pixels in the plots contained depth values outside the displayed range. The lower number of missing pixels in (c) on non-cooperative target surfaces with low reflectance, especially the mannequin's trousers, demonstrate the MCMC algorithm's advantage in resolving low-intensity returns.

range with cross-correlation, which implies the maximum values do not always correspond to the correct surface position. This is consistent with the discussion of illustrative pixel data showing the strength of the MCMC method in processing low amplitude ladar signals hidden in backgrounds in that the posterior mode is more informative, robust and reliable.

2.5.2 Mannequin Concealed by Fence: Cross-correlation and RJMCMC

In the next example, a wooden fence is placed approximately 1 meter in front of the mannequin, as shown in Figure 2.14. The image resolution of the scanned upper half mannequin is 32 by 48 pixels. Because of the area of the laser footprint, it is highly likely that some pixels may observe multiple reflections composed of some or all of the fence, the mannequin, and the pillar behind, where the beam hits occluding boundaries. In this situation, determination of the number of surfaces is an additional crucial issue and so we apply the RJMCMC method to obtain varying-dimensional ladar signal analysis.

In one sweep of the RJMCMC algorithm, the fixed-dimensional parameter updates (steps 1-3 of Section 2.2.4.1) follow the MCMC sampler settings. Jumps between parameter subspaces with different dimensions are accomplished in the same manner as defined in Section 2.2.4.2. Although our expectation would be that the number of surface returns in any single pixel would not be greater than three in this example, we are conservative in allowing the varying dimension sampler to explore k values from 0 to 5.

Figure 2.15 illustrates representative pixels containing zero, one (either mannequin or fence), two (fence and mannequin) or three returns (fence, mannequin and pillar), with the corresponding photon counts histogram, unified cross-correlation values, $p(k|\mathbf{y})$ estimates and fitting results from the RJMCMC sampler.

(a)

(b)

Figure 2.14: Close-up photograph of the upper half of the mannequin positioned at 1m behind a wooden fence. The scene was scanned at a stand-off distance of 325m in daylight.

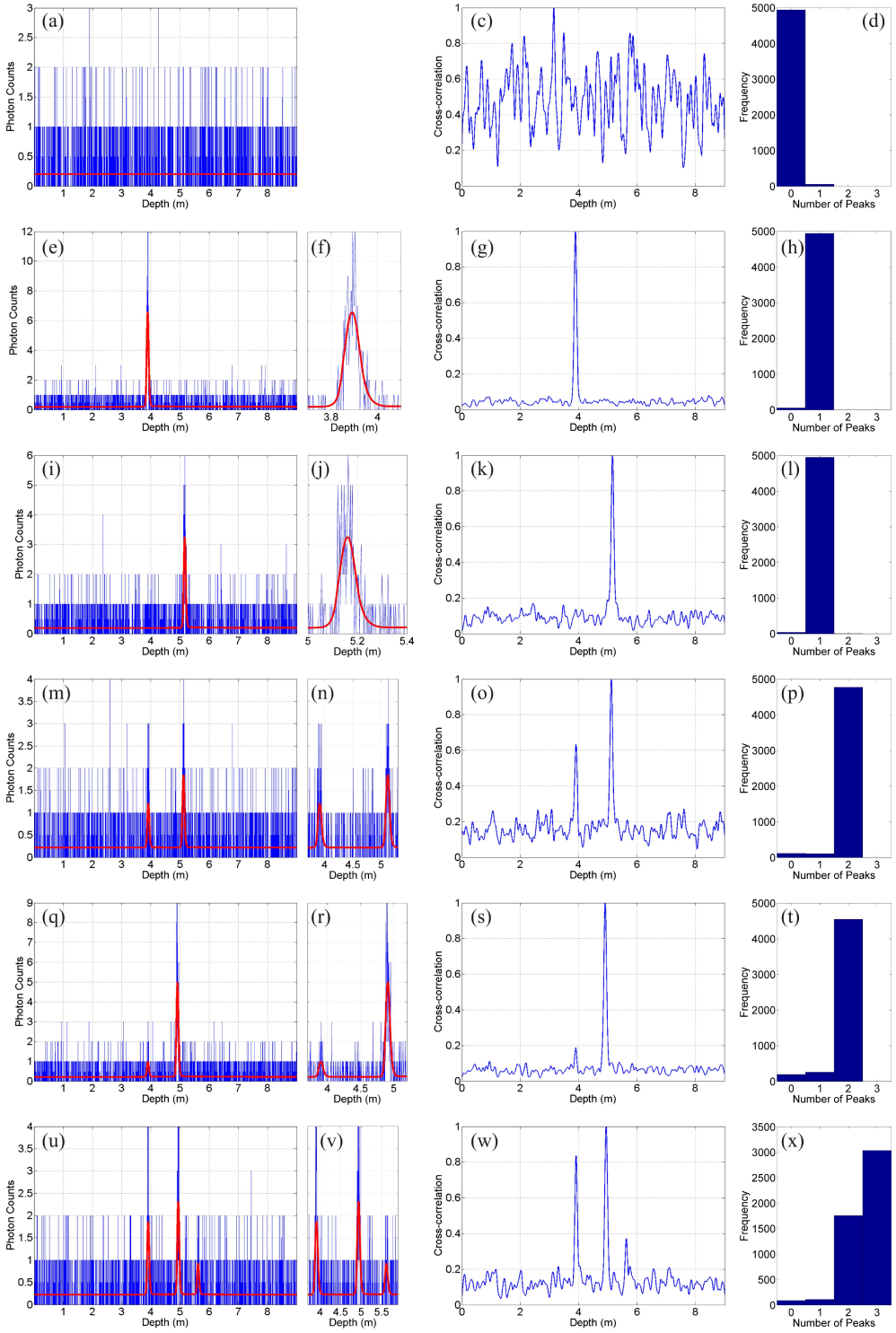
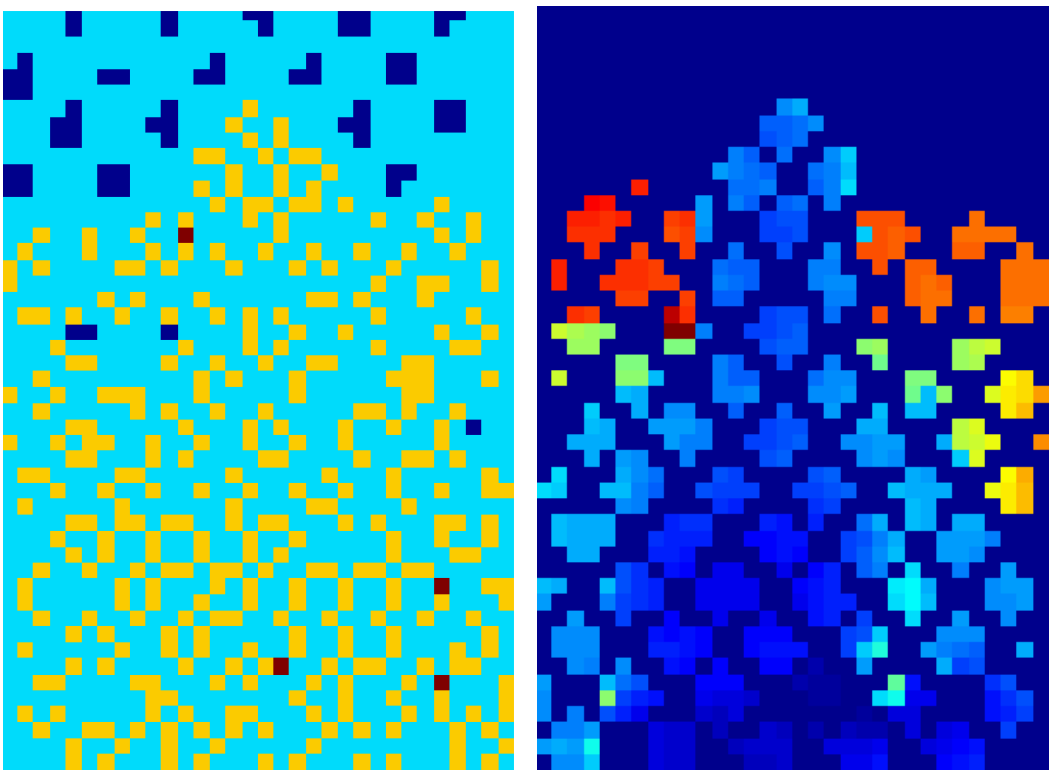


Figure 2.15: Comparison of LaDAR signal analysis for the concealed mannequin using RJMCMC and cross-correlation. The first column shows the raw data and the posterior parameter estimates from the RJMCMC method, while the second column gives the magnified plot of the signal peaks. The third column shows the cross-correlation function. The right hand column shows the posterior probability estimate of the number of surface returns, $p(k|\mathbf{y})$.

The first row of Figure 2.15 illustrates a pixel in which the beam misses all three targets, so that no surface return exists. The use of the cross-correlation method is difficult when there is no surface return as shown in Figure 2.15(a) to 2.15(d). In comparison with Figure 2.12(g) from a small signal-to-background ratio pixel, Figure 2.15(c) shows the probable existence of at least one surface return. However, according to the asymptomatic posterior probability estimate of $p(k|\mathbf{y})$, no target return is the most probable conclusion. If we examine the second and third rows of Figure 2.15 then we see the situations analogous to Figure 2.12(e) in that there are single returns from fence and mannequin respectively. The difference in this case is that we have applied full RJMCMC chains, so that the posterior probability estimate, $p(k|\mathbf{y})$, shows one return.

Of more interest are those pixels containing more than one return, shown in Figure 2.15 (m)-(x). The fourth row has distinct returns from the fence and mannequin, and the RJMCMC sampler has a very strong preference for two returns. The fifth row is far less distinct, but the sampler again shows a strong posterior probability estimate of two peaks, although the second one might be difficult to detect automatically on a cross-correlation detector, e.g. using a fixed (or even proportional) threshold. Due to the varying surface reflectance and angles, pixels can have different photon intensities, which makes it a difficult problem to choose a reliable threshold. The corresponding parameter estimates of the two surface returns shown in Figure 2.15(q) correspond in depth to the known ground truth of the relative separation. Finally, the last row shows one of the pixels in which the beam partially reflects from the fence, partially transmits through a gap and hence reflects from the mannequin, but near an occlusion boundary so that part reflects from the pillar behind. The posterior estimate of k favours 3 surfaces but it is by no means as clear cut as the earlier examples, and the parameter estimates of the 3 surface positions shown in Figure 2.15(u) correspond to the fence, mannequin, and pillar separations at this point.

To better illustrate the posterior estimates of the number of surfaces, $p(k|\mathbf{y})$, Figure 2.16(a) shows those pixels in which $0, 1, 2$ and 3 surfaces were estimated. Physically, one expects



(a) (b)

Figure 2.16: (a) Map of for different k values: $k = 0$ in navy blue, $k = 1$ in Cambridge blue, $k = 2$ in yellow and $k = 3$ in carmine. (b) Projection of the reconstructed surfaces.

no returns when the laser hits no surface, or where the surface angle is so oblique (e.g. at the extremities of the pillar) that no return is likely. In this image, these are primarily where the beam goes through the fence but above both mannequin and pillar. When $k = 1$ it hits a single surface, and when $k = 2$, two surfaces, as described above. There are only a few pixels for which $k = 3$, where the beam grazes the left arm, and no estimates of $k > 3$. Figure 2.17(c) shows a surface plot of the meshed (x, y, z) data for the 3D image of the partially concealed mannequin behind the fence. As the mannequin surface has been interpolated and smoothed from the raw data values it should be considered as illustrative, but there was no necessity for outliers removal, and the shape of the upper body is relatively well defined.

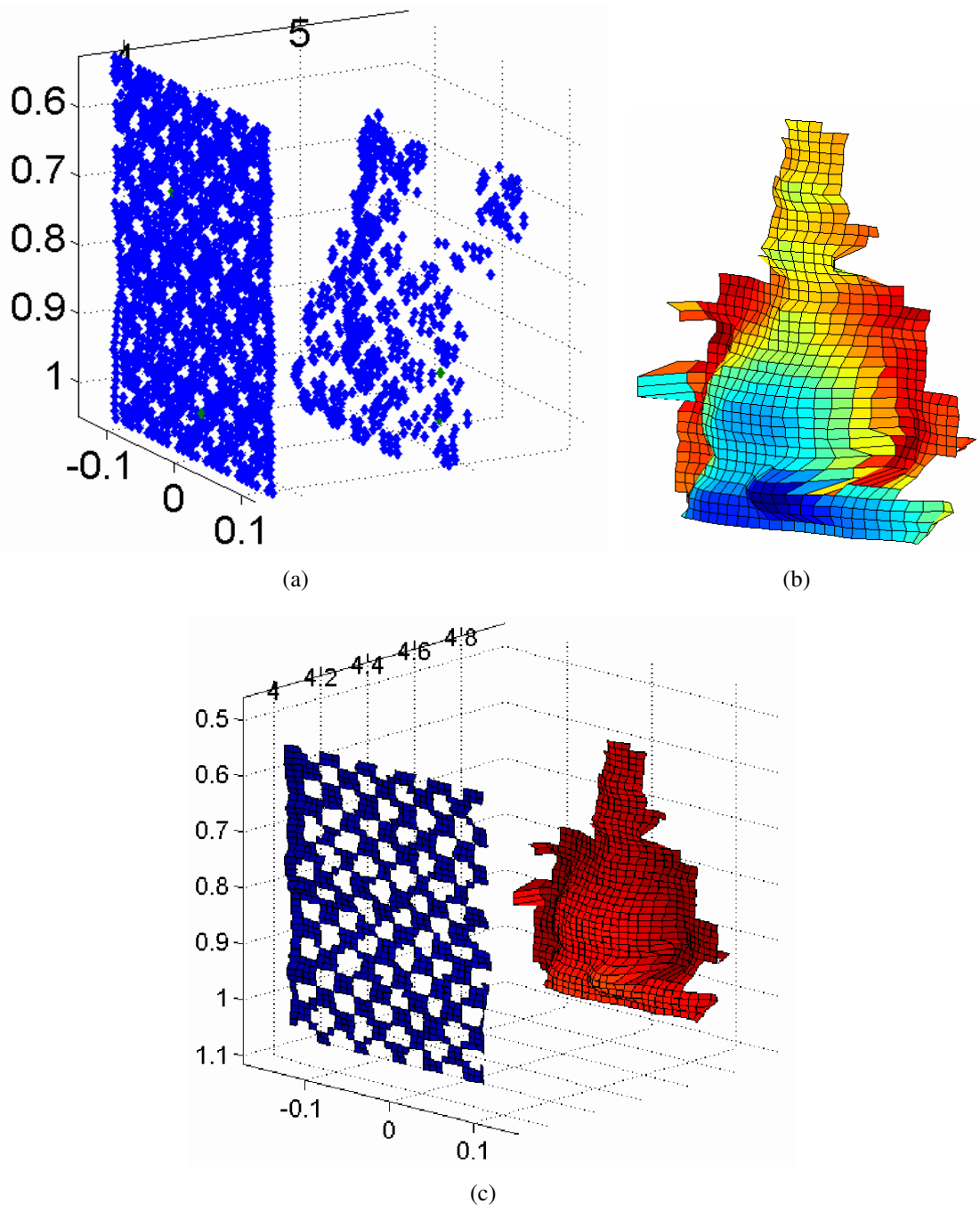


Figure 2.17: (a) Fence layer and the reconstructed 3D image of the mannequin layer. (b) Reconstructed 3D mannequin image with interpolation and smoothing technique. (c) Fence and mannequin with interpolation and smoothing.

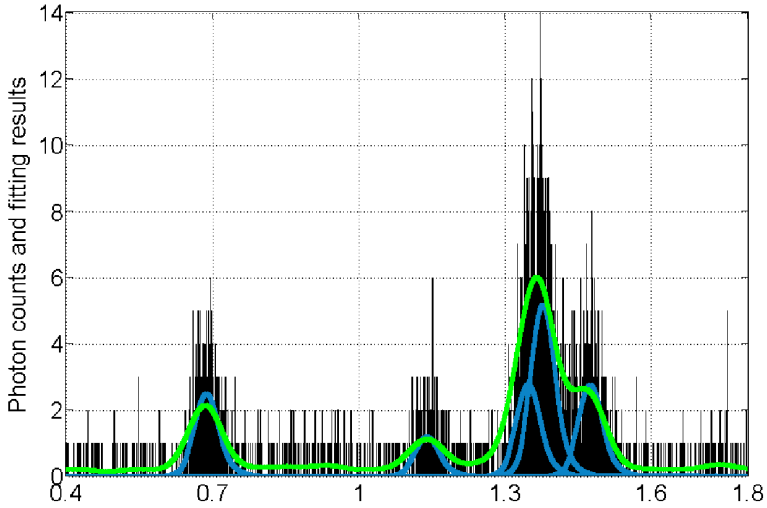


Figure 2.18: Analysis of TCSPC data from a real target containing 6 distributed surfaces with known separation distances: {450, 10, 200, 30, 90mm}. The blue line gives the 5 peaks detected by RJMCMC method with separations determined to be {452.4, 207.6, 27, 100.2mm}. The green line is the cross-correlation of the signal (for the sake of display clarity, the maximum value is scaled to be 6), which gives 4 peaks with separations {454.8, 225.6, 97.8mm}.

2.5.3 Real data with known geometry: Cross-correlation and RJMCMC

We set up a remote target at a range of approximately 325 meters, which contained 6 distributed surfaces with separations between adjacent surfaces of {450, 10, 200, 30, 90mm}. The photon counting histogram in Figure 2.18 was collected with the scanning system using a 3MHz pulse repetition frequency and 50μW average laser power, the bin resolution was 4ps. The RJMCMC sampler used here is exactly the same as the one for the fence data but allows k to vary from 0 to 10.

According to Figure 2.18, both RJMCMC and cross-correlation methods succeed in detecting distinct return signals. For the two surfaces separated at 30mm, they merge to be a single peak in cross-correlation values. In comparison, with assistance of Merge/Split updates, RJMCMC can easily separate them. However, both methods fail to distinguish

the peaks 10mm (17 channel bins) away from one another, and instead place a combined return, which results in the increased estimated distances from the combined signal to its neighboring peaks, i.e. the two peaks corresponding to the surfaces separated by 450 and 200mm.

2.6 Conclusions

In this chapter, we have provided a review of existing laser ranging and 3D imaging systems. Particular emphasis has been place on the time-of-flight (TOF) system using time-correlated single photon counting (TCSPC) technique, which offers higher measurement resolution and data acquisition speed compared to other imaging systems. We have applied the MCMC/RJMCMC techniques within the Bayesian framework to analysis the full waveform LaDAR data with fixed/unknown number of surface returns. Using a scanning sensor, and coupled with algorithmic development, we are able to detect multiple surface returns within the field of view of pixels, creating multilayer images. In addition, we have incorporated a delayed rejection step, which enables the fast movement between separated channel regions to locate the peak returns, improves the mixing performance and shortens the burn-in period. Moreover, we have reviewed the current convergence diagnostics and investigated the Gelman and Rubin diagnostic and the Heidelberger and Welch diagnostic to assess the convergence performance in the targeting LaDAR application and effectively control the chain length.

To demonstrate the method, and compare it with the thresholded correlation analysis, we have used selected data from two images of a distant target, the first in full view, the second viewed through a trellis fence. In general, RJMCMC analysis is advantageous in supplying principled estimates of both the number of surface returns and the associated parameter vectors (range, amplitude, and background level). This allows us to construct

multilayered 3D images. The methodology is effective in dealing with low amplitude returns, a few photons at maximum in a single bin. This adds to the covert capability of the sensor, aimed at detecting returns from uncooperative surfaces at medium range using a low-power source laser diode. This has application in defence and security when objects of interest may be partially concealed, or viewed through semitransparent surfaces, such as through windows.

Chapter 3

Parallel Markov Chain Monte Carlo Algorithms

3.1 Introduction

As a powerful simulation tool, RJMCMC can approximate the posterior distribution of parameter vectors with unknown dimensionality by constructing a Markov chain, which makes it particularly applicable to estimate the number of surfaces and the related parameters in LaDAR signals. Compared with the conventional methodologies for LaDAR signal processing, Bayesian inference using RJMCMC algorithms shows dramatically better performance in resolving closely separated surfaces or detecting surface returns from a comparable or even higher background level [22, 61]. However, the insurmountable obstacle of the intensive computation, with complexity $O(NkL)$, largely inhibits its application scopes. Because usually a long chain is required to fully explore the state space of the posterior distribution, especially where exist between-model mixing difficulty and high inherent autocorrelations of the output samples.

Parallel computing has become an inevitable trend to conquer the previously insurmount-

able obstacles of intensive computation. The primary objective is to speed up the simulation process and reduce the memory space requirement on individual machines by distributing the full tasks onto multiple separate processors. It holds the great potential to deal with the complex statistical model using massive numerical calculations. In particular, it offers the possibility to improve the simulation efficiency of RJMCMC algorithm for LaDAR signal analysis and overcome the restrictions on large memory storage.

Unlike the direct parallel implementation for naturally decomposable algorithms, such as matrix calculation, parallel processing of RJMCMC algorithm is not only an implementation technique in the computing domain, but also a challenging problem in the statistical domain. Because a Markov chain is serial by nature, and cannot be directly constructed in a parallel platform. It is not enough to solely take into consideration of the practical considerations for parallel computing; but, further, we should be cautious about the statistical property to provide an invariant limiting distribution, where arises the difficulty of RJMCMC parallelization.

3.2 Background of Parallel Computing

3.2.1 Why Use Parallel Computing?

Traditionally, serial computation is to break a problem into a discrete series of instructions and execute them one by one on a single computer with one Central Processing Unit (CPU) (see Figure 3.1(a)). In the simplest sense, parallel computing splits a problem into a set of tasks and executes them simultaneously on multiple different CPUs (see Figure 3.1(b)). The computer resources can include a single computer with multiple processors, an arbitrary number of computers connected by a network and a combination of both.

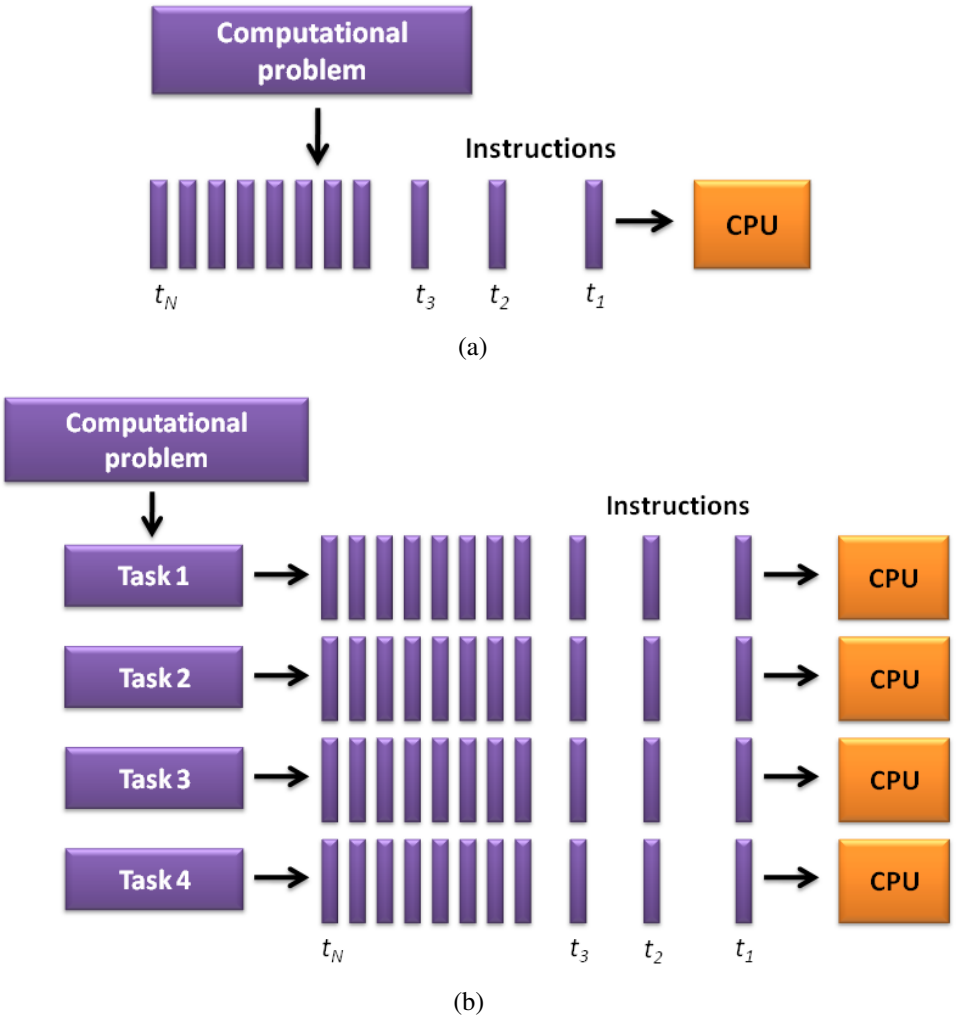


Figure 3.1: (a) Serial computation of a set of series instructions on a single CPU. (b) Parallel execution of a set of independent tasks on multiple CPUs.

Why parallel computing? The answer is simple: Because there exists a need to solve large scale problems that are so large and/or complex that it is difficult, impractical or impossible to be dealt with on a single processor, especially given the limited computer memory and processing time. Some examples include earth environment prediction, quantum chemistry, computational biology, data mining for large data set, the application of statistical models to large data set. Moore's Law [62] promises that the size of the problem researchers are interested in solving is increasing faster than the uninterrupted increasing

of the computational power of a single processor. Therefore, combining multiple processors builds a bridge across the gap. In parallel computing, the computational problems can be broken apart into independent tasks that can be solved concurrently, such that there are multiple programme instructions being execute at any moment in time. This allows the problem to be solved in less time with multiple computer resources than on a single computer resource, and each with reduced memory space requirement due to the reduced problem size.

Other reasons for using parallel computing include: first, according to Moore's Law, the trend in overall growth of the world's most powerful computer doubles approximately every 18 months. Accordingly, the size of feasible solvable problems also nearly double every 18 months. Even though a super computer can achieve some level of the computation any time soon, it is relatively much easier and cheaper to combine multiple cheap processors to work on problems with smaller scales than acquiring a supercomputer that would be as powerful on its own. Furthermore, it provides a possibility to take advantage of non-local resources by using available computer resources on a wide area network or even the Internet.

Due to the above reasons, parallel processing has come to the forefront and is the only way to solve the large computational intensive problems and reach the necessary performance on a large scale. According to [2], multiple processing has been widely used in various application areas. The share of application areas is plotted in Figure 3.2.

3.2.2 Flynn's Classical Taxonomy

There are different ways to classify the configurations of parallel processors. One of the more useful and general classifications is Flynn's Taxonomy [63], introduced in 1972. Flynn distinguished the multi-processor computer architectures according to how the in-

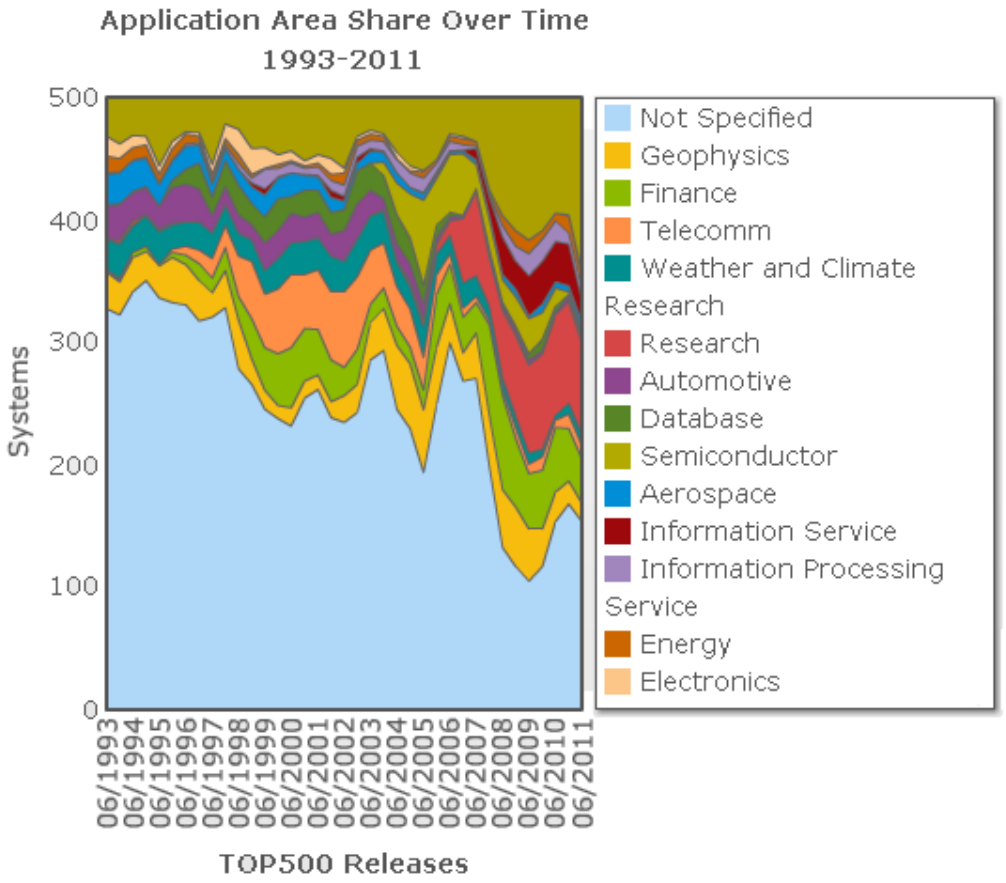


Figure 3.2: Application area share of top 500 high performance computers using parallel processing systems (from [2]).

formation processing takes place inside of a computer, which is defined along two independent dimensions, the data stream and the instruction stream. Both concepts have a possible state, either Single or Multiple. Accordingly, the computer architectures are classified into four types, as displayed in the matrix shown in Figure 3.3, while the illustrations are given in Figure 3.4.

- SISD - Only one instruction stream is being executed on the CPU for only one data stream input during any one clock cycle. It includes the most common type of serial computers using the von Neumann architecture [64].

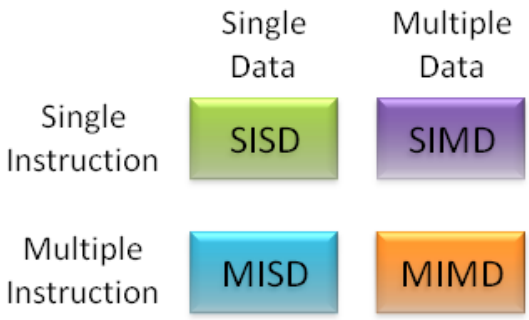


Figure 3.3: Flynn's taxonomy.

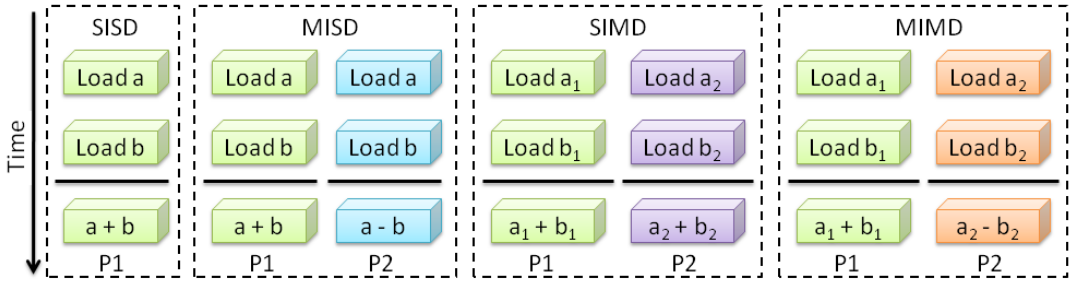


Figure 3.4: Illustrations of SISD, MISD, SIMD and MIMD computer architectures in Flynn's taxonomy.

- MISD - Multiple processing units operates on the same data stream independently via separate instruction streams. It describes various special computers but no particular class of serial or parallel computers.
- SIMD - All the processing units act on the same instruction stream but for different data fed into each processor. It is an architecture of parallel processor "arrays", which is frequently used in most modern computers, particularly those using graphics processor unit (GPUs).
- MIMD - Every processing unit may be executing a different instruction on a different data stream. Many MIMD architectures include SIMD components.

Most of the parallel system falls into the categories of SIMD and MIMD. SIMD-based parallel systems usually consist of a relatively large number of weak processors, each connected with a relatively small memory. As shown in Figure 3.5, the processors are configured as a matrix-like topology, and from which comes the name of processor array. The processor elements are linked to the control processor, which conducts the program compiling and manages the processor array. The prime advantage of the SIMD computers is that the processors work synchronously, which is convenient for program tracing and debug. Moreover, this architecture is best suited for specialized problems characterized by a high degree of regularity, such as graphics and image processing. However, for unstructured problems, more general problems with high level of flexibility of data manipulation or requiring data transfer between processors, it may not an appropriate choice.

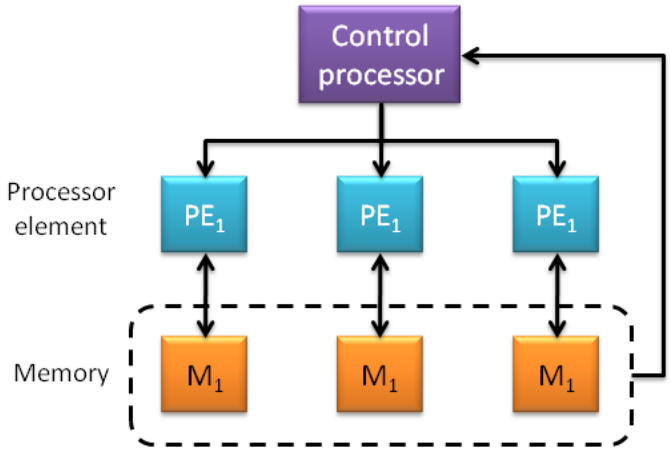


Figure 3.5: Processor array in SIMD.

MIMD-based parallel systems are defined by the memory architectures: shared memory and distributed memory system. The division is conceptualized in the terms of the connections between the processors and memory. This is discussed next.

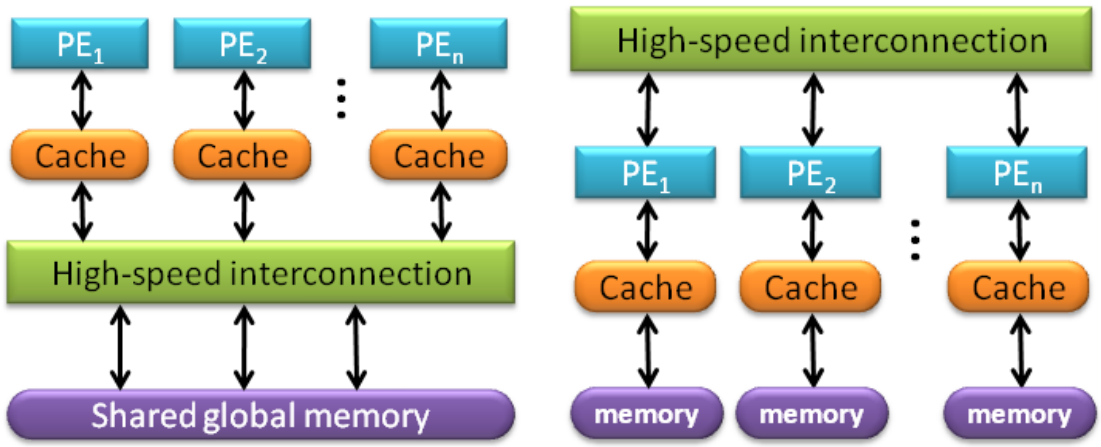


Figure 3.6: Memory architecture for MIMD parallel computers: (a) Shared memory module. (b) distributed memory module.

3.2.3 Parallel Computer Memory Architectures

Shared memory computers

Shared memory parallel computers are characterized by a number of processors (processing element PE) connected to a global (main) memory via a high speed bus. They are equipped with the cache memory, which is a relatively smaller and faster memory subsystem used by the central processing unit to increase the data throughput and reduce the average time to access main memory. It is inserted between the processor and the main memory to store both the data and instructions predicted to be used next. A common memory module is given in Figure 3.6(a).

The multiple processors of this type can operate independently but share the same memory space. Hence, the change of the memory locations caused by one processor will affect all other processors. This gives the main benefit that is the fast and uniform data sharing. Unfortunately, this advantage comes at a price. Users should be cautious about synchronization constructs and ensure the correct access of the global memory. Another

disadvantage is that due to the bandwidth limit of the shared memory-CPU path, it is difficult to scale the shared memory architectures to more than 32 processors.

Distributed memory computers

In the configuration of distributed memory computers, each processing unit has its own local memory and operates independently. Unlike the shared memory computers, there is no global memory across all processors. A cluster of stand-along workstations are interconnected by a structured network. Typical network topologies are meshes, toruses and hypercubes. To allow one processor to access data in another processor, processors communicate via formal message passing. One popular library established for standard interface is the Message Passing Interface (MPI) [65], which is convenient for programmer to develop portable applications and hence becomes our preference.

The code for distributed memory is relatively more difficult and complicate. Unlike in a shared memory architecture, the synchronization between processors is the task of the programmer. The programmers need to explicitly define how and when data is communicated. However, distributed memory architecture has a large scalability with the number of processors. Considering the flexibility and portability requirements, we will design our algorithms based on the distributed memory system.

3.2.4 Performance Evaluation

The performance of the parallel algorithm is usually measured by *speedup* or *efficiency*. *Speedup* refers to how much the parallel algorithm runs faster than a corresponding serial algorithm, defined as

$$\text{Speedup}(p) = \frac{T_{\text{serial}}}{T_{\text{parallel}}(p)} \tag{3.1}$$

for p processors. A linear speedup is obtained when $\text{Speedup}(p) = p$, and the super linear speedup occurs when $\text{Speedup}(p) > p$, which rarely happens unless using the different

memory hierarchies of a modern computer, or reducing the task size from the parallel tolerate algorithm (i.e. the algorithms supports parallel processing).

There are several methods to specify the serial execution time, which results in different definitions of speedup. In our experiment, we apply real speedup and relative speedup. Real speedup compares the parallel execution time with the fastest serial algorithm's execution time on a single processor from the parallel machines. In the circumstance where the problem size and computation complexity in the parallel tolerable algorithm is not exactly equivalent to the serial algorithm, we can use relative speedup to analyze the performance, where the serial time is defined as the total execution time of the parallel algorithm running on a single processor of the parallel machines.

Efficiency estimates the utilizations of the processors in solving the problem, relative to the wasted processing powers in communication and synchronization. A direct understanding of efficiency is the speedup achieved in each processor,

$$\text{Efficiency}(p) = \frac{\text{Speedup}(p)}{p} = \frac{T_{\text{serial}}}{p \times T_{\text{parallel}}(p)} \tag{3.2}$$

It is typically a value between zero and one, while algorithms with linear speedup or running on a single processor have an efficiency of 1.

We choose the wall clock time for timing analysis, which measures the elapsed time from the start to the completion of the task. Different from the user time, it contains the cost of the communication channel delay and programmed delay. Although wall time gives a smaller speedup, it can characterize the algorithm performance in the real computation environment.

3.3 Practical Considerations

For the parallelization of Markov chain Monte Carlo methods, we need to consider the statistical properties of the parallel strategies besides the issues related with parallel computing. In principle, parallel strategies can not be isolated from the serial MCMC/RJMCMC algorithms [66], because: first, as the prototype of parallel computation, if the MCMC/RJMCMC algorithm itself lacks expected simulation performance in terms of computation complexity and estimation accuracy, even though employing the advanced parallel techniques and well configured hardware, it is conceivable that it is difficult to achieve further speedup. Second, since parallel computing is usually problem dependent, characteristic properties of the MCMC/RJMCMC algorithms play the decisive role when determining parallel strategies, that is, different algorithms would always be parallelized in different ways. To give an example, it may be reasonable to generate multiple realizations for a Markov chain with good mixing and short burn-in period; by contrast, if mixing is poor and burn-in time is long, the sensitive solution is to parallelize complex calculation involved in a single chain or develop more sophisticate MCMC/RJMCMC algorithms to improve the mixing performance and then implement in parallel.

Considerations for serial MCMC/RJMCMC algorithms, such as mixing performance and convergence rate, can also be applied to parallel MCMC/RJMCMC. However, since MCMC/RJMCMC parallelization adds additional layers of complexity compared with serial MCMC/RJMCMC and its variants, it brings in extra issues and trade-offs for parallel algorithm design, which will not only affect running efficiency but also relate to the estimation precision of the posterior distribution for Bayesian inference. These considerations will form the foundations for parallel MCMC/RJMCMC examination and evaluation in subsequent sections.

3.3.1 General Considerations for Parallel Computing

First, there is an *unreliability* problem. In a multi-processor environment, computers may not be always reliable. They may malfunction, power off, run at abnormal low-speed or maybe the network is not operating properly. Computer failures may be independent or dependent with assigned tasks termed as independent or dependent failure. They could seriously affect parallel computing performance. On the one hand, when each processor is used to generate random samples separately either to expand partial MCMC/RJMCMC chain or to complete an entire chain, this could bring in deviations in simulation results from the their theoretical values due to the decreased sample size caused by computer failure. For instance, as discussed in [67], when simply generating multiple Markov chains on several processors, bias, which is the difference between estimation and unknown true values, might become larger when dependent failures occur. On the other hand, an even worse effect, the designed parallel algorithms may turn out to be infeasible especially under the situations that processors are allocated with specific different deterministic calculation tasks instead of exploring random sample spaces. In this case, if any processor breaks down for whatever reason, the whole processing would be terminated or stagnate. However, it is not impossible to solve these problems. Generally, according to [67], if a computer does not report back correctly within a prior specified reasonable time, the sub-task allocated to that processor will be re-computed by a properly operating machine with the same pseudo-random number seed to duplicate the exact computation. Although correct estimation results can be then obtained, longer computation time is an extra expense.

Second, in parallel implementation, *communication* between processors on distributed computers is slower than processing because of the complex procedure, dynamic changing traffic load and network speed. Thus, it is one of the principle factors that affecting speedup and efficiency. This is consistent with the fact that simulation speed may begin to slow down dramatically when the number of processors increases to a specific value since

this could bring the extra communication overhead. From this point of view, it is necessary to determine the optimal number of computers and this may change with parallel algorithms and operation environments. Moreover, another issue raised here is the *granularity* which is the ratio of processing time and communication time. Sub-tasks should be sufficiently large to avoid fine-grain processing where time spent for communication is longer than for computation. Last, although communication is inevitable, at least to some extend, it is practical to design approaches with relatively lower communication frequency. In fact, this concentration has been placed in some existing algorithms.

Third, since a single low *speed* computer may play a decisive role in determining the task completion, individual processor speed might be crucial. The ideal case is that coordinate machines will work at approximately the same speed. However, it is sometimes unachievable in practice due to different computer configuration and heavy user load. In particular, strict synchronization between cooperation processors may be involved in some parallel algorithms, and therefore idle time may exist. Since the master processor may not able to process further computation until all sub-task results are available, processors might have to wait for each other until they all reach the designated point when there is data exchange among processors at intermediate stages. Now that speed is not under control, the helpful solution might be developing *load-balancing* approach, which is basically allocating larger amount of work to higher functioning machines to seek minimum execution time and optimal resource utilization. More generally, it is to shorten the idle time that a processor stands until the last one catches up. However, it may be difficult to predict computation speed in advance. Thus, one desired approach is *dynamic loading-balancing*, which chooses task size on-line with regard to computer throughput, see [67] and references therein. Another method is to adopt *queuing approach* as presented in [68]. The entire task is divided into numerous sub-tasks and placed in a queue with order. Once a processor finishes its current sub-task, it is assigned with the next one from the front of the queue. Since the number of elements in the queue is significantly larger than processor, automatic load-balancing is carried out. Nevertheless, it fails when sub-tasks

are not independent or communication time is larger then sub-task proceeding time.

3.3.2 Special Considerations for MCMC/RJMCMC Parallelization

First of all, the *burn-in* period plays a key role when truncating the Markov chain to obtain samples under equilibrium states. In terms of parallel MCMC/RJMCMC, there might be additional difficulty to determine the burn-in time since different processors may hold different transient times when they are employed to generate a whole chain or several shorter chains. Additionally, compared with a single serial computer, if the burn-in period is not relatively short, running a chain on separate processors may not even be as efficient as nonparallel implementation since burn-in happens in all the shorter chains and results in a large number of wasted samples. Hence, from a practical perspective, a long burn-in time is an obstacle in effectively utilizing available hardware in parallel environment. Furthermore, different machines could introduce different burn-in biases due to various sample sizes [69] and convergence diagnostics. Recently, efforts have been mainly concerned to address these subsequent problems. The direct way to handle this is to predefine a unified, fixed burn-in time. However, there is a trade-off in that if the burn-in length is chosen to be too large, wasted computation will degrade parallel performance; by contrast, if it is too small then larger bias will be introduced, especially in slow-mixing chains. The improved approach is to determine burn-in time dynamically using convergence diagnostics based on sample runs, which is similar to sequential implementations. Diagnostic bias is sometimes unavoidable, see [67] and references therein. Beside this, multiple-run computer diagnostics [70] would be applied in this parallel multiple chains situation. Another approach is to analysis theoretical burn-in times; however, they are not easy for simulation in general. The safest approach as suggested in [67] is to initialize Markov chains with perfect samples [71, 72, 73] and then generate updates afterwards using proposal distribution. But perfect simulation is complex; thus, inefficiency might be a drawback.

Moreover, a special concern in parallel MCMC/RJMCMC is that whether the *stationary distribution* would converge to the unique target distribution supposing the chain is sufficiently long and ignoring unreliability and burn-in bias. Dissimilar to serial MCMC/RJMCMC, the appropriate limiting direction is not always maintained and this arises from several certain reasons. First, MCMC algorithms are normally subject to detailed balance which is stronger than irreducibility and periodic to guarantee the invariant distribution. However, as the condition of detailed balance, reversibility may not always be held especially when sample states swapping occurs between separate Markov chains with distinct transition probability, for instance, the parallel Metropolis-coupled MCMC (introduced later in Section 3.4.2.2). Furthermore, in some cases, observation space or parameter space are segmented such that intensive computation could be shared among multiple processors. Consequently, states moves might be constrained within a specific domain. As a result, invariance is not in support of irreducibility since sample generation is only under partial randomness in terms that only a fraction of state space is under exploration. Finally, parallel processing is applied by dividing a task into multiple sub-tasks and executing concurrently on different computers. Note that this only works when none of the sub-task relies on the results of any others. Nevertheless, some statistical models, such as the Latent Gaussian process, may hold tight correlations in either parameter space or data space and therefore careless partitioning may destroy this relationship which would in turn result in incorrect limiting distribution. In sum, the preferred MCMC/RJMCMC parallelization should be efficient and meanwhile not disturb the consistency of sample path average.

Furthermore, *variance* is used to describe the scale of how the simulated value is spread out from its expected value so as to capture the statistical dispersion of Monte Carlo averaging. It is an overall index affected by various factor such as sample size, unreliability, independence among generated sample and low mixing. As for parallel Markov sequences, one valid approach is to reduce the correlation of random number streams on separate processors by assigning identical random number seeds to each machine. Al-

though dependency could not be completely removed since only pseudo-random number could be produced in computer simulation, at least to some extend, it helps to reduce the degree of sample overlapping. Technically, there have already been a couple of methods to serve this requirement [74, 75].

Generally, the potential for parallelization of any algorithm can be analyzed by dividing the algorithm into parts that can be executed in parallel and into those that have to be executed sequentially. According to Amdahl's Law [76], the theoretical maximum speedup is bounded $1/(1-f+f/p)$, where f is the fraction of the program that can be parallelized and p the number of processors. The bottleneck of parallel performance is the serial computations. As illustrated in Section 2.2.5, the MCMC and RJMCMC approaches involve intense computations per iteration (where each iteration produces a sample). In the basic algorithms, samples are generated sequentially, following the Markov property. Within the iteration, the moves are performed in a sequential order, relying on the outcome of the previous move. Although the sampling scheme is sequential, the computations, such as the likelihood computation, are parallelizable, which exploits the fine-grain decomposition for parallelizing within iterations. Since the fraction of this part is different in different move types, and its computational complexity is not fixed but increases with the problem size, the expected parallel performance assessed from Amdahl's Law is data dependent. In addition, we will introduce some parallel algorithms in Section 3.4, which make use of the conditional independence of the underlying statistical model and parallelize the sampling procedure to a different extend, providing the coarse-grain decomposition. The different granularity determined by different parallelization technique affects the fraction of the parallelizable part of the program. Therefore, the expected maximum speedup is also algorithm dependent.

3.4 Current Algorithms for MCMC Parallelization

This section will introduce current parallel MCMC algorithms for fixed dimensional problems. As shown in Figure 3.7, these methods can be classified into three categories: parallel generation of a single chain, parallel generation of multiple chains and combinations of both.

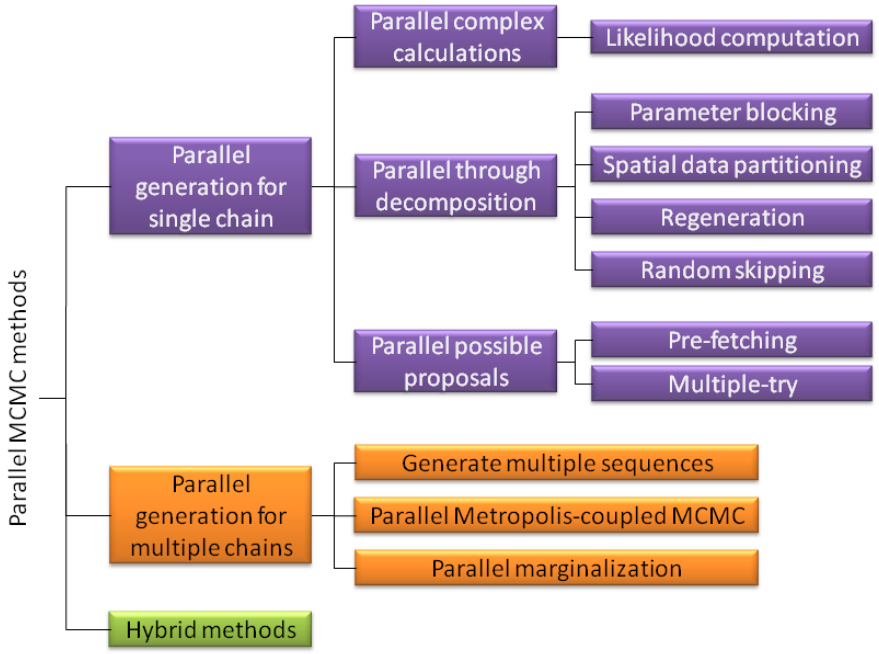


Figure 3.7: Summary of current MCMC parallelization algorithms.

3.4.1 Parallel Generation of A Single Chain

3.4.1.1 Parallel Local Complex Calculation

With reference to M-H kernels (Section 2.2.3.3) and Equation (2.26), computations in MCMC algorithms come from the acceptance ratio. For the sake of implementation convenience, proposal distributions are usually chosen to be standard distributions, which

are easily sampled from. Normally, the prior probabilities are in a simple mathematical form and can be quicker evaluated. In the Bayesian framework, the target distribution is the posterior distribution of the parameter vector with fixed dimensionality, wherein the likelihood function contributes the majority of the computation task, particularly for large data sets. Therefore, the direct solution for parallel processing is to distribute the complex likelihood computations among processors. This is a safe method that does not affect the statistical properties of the serial sampler and hence protects the invariance of the generated sequence.

3.4.1.2 Parallelisation Through Decomposition

Regeneration

One conceptually straightforward scheme for low-dimension state-space problems is to implement the regeneration technique for MCMC [77, 78] in parallel. Regenerative simulation is basically dividing a single Markov chain by a pre-chosen state point θ^* into several individual segments referred to as *tours*. Each tour will begin with and terminate at this point. Since these tours are independent and identically distributed, they can be generated simultaneously on separate processors and finally joined together to form a long chain.

As discussed in [79, 80, 81], this method can help to solve the burn-in problem and construct better parameter estimates. However, the assumption of a non-zero probability for returning back to its original starting state restricts its practical applicability to a low-dimensional discrete state space or a continuous space with a revised algorithm [78] for non-zero re-entry probability. Even in the low dimensional case, the re-entry proposal distribution should be carefully chosen in order to improve mixing performance caused by an inappropriate transition kernel and multimodality problem, as suggested in [78]. Following this, regeneration time identification is still a tough task although methods have been

proposed in [79, 82, 83]. It is actually a more significant issue in the parallel environment to achieve load-balancing unless the chain length can be predicted.

Blocking

For high dimensional MCMC problems, one tempting approach evolves from the blocking scheme originally designed in serial MCMC to improve mixing performance [84]. The basic idea is to decompose the state-space and group together its components into a number of manageable blocks that are updated concurrently on separate processors. Unfortunately, this technique is not always valid in practice because partial correlations usually occur among these blocks as in the example given in [68]. Therefore, to preserve a correct equilibrium distribution, blocking must be carried out under the conditional independence condition as described in [66].

There, the parameter vector is modeled as $\phi = (\delta, \theta)$, where δ represents the parameters of direct inferential interests and θ reflects other details which can not be directly observed. Typically, the δ update can be much faster than the θ one; hence, it is beneficial to speed up the algorithm by parallelizing the θ computation. Suppose the components of the chain are $(\delta_1, \delta_2, ..., \delta_p; \theta_1, \theta_2, ..., \theta_q)$, the general form of the blocking scheme explicitly presented in [66] is: partitioning $(\theta_1, \theta_2, ..., \theta_p)$ into $T = \{T_1, T_2, ..., T_c\}$ where T_i is a set of mutually independent blocks giving the remaining blocks. Blocks in T_i can be updated simultaneously in parallel while T_j might depend on the value of T_i and then updated in turn. After θ is completely updated, δ can then be processed. The above scheme can also be employed when the p components of δ are not conditionally independent. Several versions for simpler parameter structure are given in [66, 68].

In spite of possible better mixing, the obvious limitation of blocking as discussed above originates from its tight independence requirements, which are not always applicable and hence bring difficulties when partitioning the state-space. Besides, the massive communications overhead could dramatically degrade parallel performance when collecting

summarized statistics from each individual block in T_i to compute T_j and passing the current states of θ to the root computer for updating δ and vice versa. Also, this message passing scheme is built on synchronization in that each processor must reach the same updating point before moving on to the next step. Therefore, this algorithm is sensitive to load-balancing and might not scaled well with the number of processors.

Spatial partionning

To overcome these issues, a *spatial partitioning* approach is proposed to construct a partition scheme T in the observation data domain rather than the state domain (e.g. blocking) and then allocate the blocks to a number of processors in a way to minimize inter-processor communication. This algorithm is investigated in [85] in the context of a latent Gaussian model with Poisson count data; however, it could be adapted and then applied to other problems, especially those involving intensive computation resulting from large set of observed samples for Bayesian inference.

The parallelization strategy for spatial partitioning is to decompose the observation space into MK subsets, which are then allocated to M groups. That is, each group contains K subsets denoted as S_{mk} (i.e. the subset k from group m) and points in these segments are assumed far enough apart to satisfy a threshold distance D as below for small inter-point correlation:

$$\min_{k_1 \neq k_2} \min_{s_i \in S_{mk_1}; s_j \in S_{mk_2}} d(s_i, s_j) \geq D \tag{3.3}$$

Hence, these K subsets can be sampled in parallel on condition of surrounding subsets from remaining groups. The example in [85] helps to illustrate the partitioning scheme [86] for a two-dimensional lattice data space. The best performance for the laser-tissue models with three different material properties is a speedup of 6 with 8 processors. A similar simulation for a chemical system with short-range intermolecular forces is introduced in [87].

The computational expense for each processor is reduced through this location portion-

ing approach, such as in this latent Gaussian model, where the whole covariance matrix is divided into several small matrices. However, practically, obtaining an exactly equal number of points in the same-group subsets and neighboring subsets is unlikely to occur, which will in turn lead to an unbalanced load distribution. Consequently, the optimal subset size and number of processors should be carefully selected to get better simulation performance. The other problem of this sampling scheme is that it would only produce an approximation to the target distribution unless subsets within each group are completely independent. In other words, correlations are weakened when subsets are assumed and treated as independent when updating concurrently in parallel even if the correlation distance D is sufficiently large. Further, subsets may not only rely on surrounding segments but also all other subsets. To address this issue, a solution is suggested in [85] which samples from the full conditional distribution on the whole set of data values including surrounding subsets. Its negative effect is that the processor idle time will probably be increased when simultaneous updating ability is lost.

Random skipping sequential (RSS) MCMC

It has been shown that sequential updating is crucial to accelerate Monte Carlo simulation and yield a faster convergence rate through domain decomposition. This is feasible for parallel implementation [88]. However, strict detailed balance is not always obeyed since only the moves in active regions are reversible or, as discussed above, the full conditional distribution might be replaced by a partial conditional distribution, which will consequently compromise the precision of the equilibrium distribution. Fortunately, thanks to an improved parallelization version investigated in [87], termed the random skipping sequential (RSS) Monte Carlo algorithm, not only the correct stationary distribution is preserved, but inter-processor communication and unsynchronization are reasonably reduced.

The proposed approach is based on spatial domain decomposition while introducing the sequential updating concept with the crucial improvement of consideration about move

type dependency. The spatial domain is firstly divided into stripes with one processor responsible for one stripe; following this, moves of the same type from different stripes are independent and thus isolated for parallel implementation; after that, executing a different move type in turn; finally, when the whole simulation box has finished updating, information at the border of the sub-domain is exchanged for the next round of updating. The algorithm is evaluated on a two-dimensional lattice gas model, which achieves a highest speedup of 9.5 with 20 CPUs.

The periodical communication scheme could dramatically reduce the frequency of message passing and the unsynchronization, which helps to substantially reduce simulation time for both moderate and large size systems. Meanwhile, by identifying the form of the parallel transition kernel, the detailed balance is verified [87]. However, the independence property among move types supposed in this algorithm does not always exist in reality. Similar to the difficulty faced by the blocking approach, the parameter space might be too complex to decompose. Further, correlation of the same move type in different sub-domains is ignored. This is probably under the assumption that this two-dimensional system is mutually correlated in a short spatial range. As discussed in spatial partitioning, this condition is sometimes unattainable.

3.4.1.3 Parallel Possible Proposals

Pre-fetching

In general practice, the main computational burdens are attributed to likelihood computation since the proposed values are generated immediately from the proposal distributions with a pre-chosen simple density function and the prior densities are evaluated promptly. Hence, it is necessary to accelerate MCMC simulation by reducing the time spent in likelihood evaluation. *Pre-fetching* is such an algorithm designed to meet this requirement [68].

This algorithm is carried out on the basic of a tree structure where each node represents one possible sample value and the full tree characterizes the full set of possible outcomes in h updating steps, as illustrated in Figure 3.8. Suppose the current state at time t is X_t and regard it as a *parent* node in the tree. Starting from this point, at time $t+1$, X_t has two children standing for two possible values X_{t+1}^1 and X_{t+1}^2, while the child along the right branch (i.e. X_{t+1}^2) is always the proposed sample. Repeating this, the whole tree is fully constructed and the total number of leaves at the bottom level is 2^h for h sequential iterations. Interestingly, it can be easily found that these 2^h values have already captured all the possible outcomes in the tree from its root to leaves.

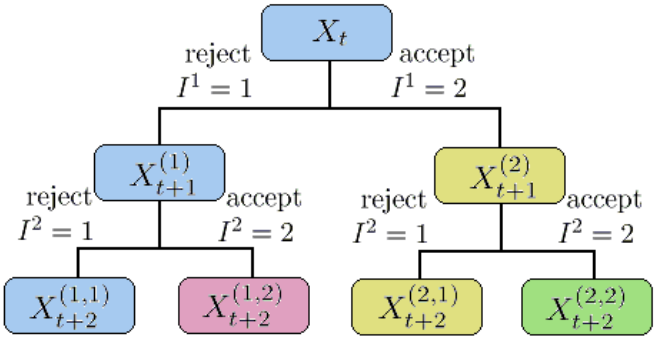


Figure 3.8: Leaves at the bottom level capture the full set the possible outcomes. When $h = 2$, there are 2^2 unique values corresponding to four unique likelihoods. Nodes in the same colour have the same value, that is, children in the left branches have the same value as their parents.

For the sake of clarity, some notations are introduced for the general case:

$$X_{t+h}^{(I^1, I^2, \ldots, I^h)} \tag{3.4}$$

where X_{t+h} denotes the state at time $t+h$ and I_j indicates whether a proposal is rejected or accepted at the j^{th} step:

$$I^j = \begin{cases} 1 & \text{reject and then left branch} \\ 2 & \text{accept and then right branch} \end{cases} \tag{3.5}$$

Let $Z_{t+j}^{(I^1,...,I^j)}$ denote the proposal value generated at time $t+j$ and $(I^1,...,I^j)$ indicates its evolution path. The acceptance/rejection procedure is defined as:

$$X_{t+j}^{(I^1,...,I^j)} = \begin{cases} X_{t+j-1}^{(I^1,...,I^{j-1})} & I_j = 1 \\ Z_{t+j-1}^{(I_1,...,I^{j-1})} & I_j = 2 \end{cases} \tag{3.6}$$

The predominance of the pre-fetching algorithm is that it produces the whole set of likelihoods for h steps within the time required for one likelihood evaluation by following the steps below:

- Step1: Construct a full tree for all the possible states within h updating steps as presented above.
- Step2: Assign 2^h processors to compute the target densities (including likelihoods computation) for all of the unique possible values of $X_t \cup \{X_{t+j}^{(I_1,...,I_j)} : I_j = 2\}, j = 1,...,h$.
- Step3: Travel down the tree; identify the corresponding target densities and determine the realization i^j of I^j obeying the standard Metropolis-Hastings rule:

$$i^j = \begin{cases} 1, & \text{with probability } 1-\alpha \\ 2, & \text{with probability } \alpha \end{cases} \tag{3.7}$$

where

$$\alpha_j = \frac{\pi(z_{t+j}^{i_1,...,i_{j-1}}) P_t(x_{t+j}^{i_1,...,i_{j-1}} | z_{t+j}^{i_1,...,i_{j-1}})}{\pi(x_{t+j}^{i_1,...,i_{j-1}}) P_t(z_{t+j}^{i_1,...,i_{j-1}} | x_{t+j}^{i_1,...,i_{j-1}})} \tag{3.8}$$

The algorithm is implemented on the fractionally integrated autoregressive moving average model, which gives a speedup of 6 with 32 processors. The benefits of this algorithm are: first, within one *time unit* required for one likelihood computation, the target distributions of all the 2^h possible states are evaluated for h iterations, which leads to a speedup.

This algorithm is especially ideal for problems with a long burn-in period. However, this approach is not particularly efficient owing to its inherent computational waste given that only one of the 2^h paths is eventually chosen. As a result, the sampling rate is only increased by $log_2(p)$, which is far from linear speedup. In spite of running efficiency, this scheme is nearly applicable to all MCMC models without any constraint on *independency* and *dimension size*. Thus, it provides an alternative approach when all of the above methods are not feasible. However, the tree structure may be modified and then this method probably becomes more complex for more sophisticated statistic problems where various move types occur.

3.4.2 Parallel Generation of Multiple Chains

3.4.2.1 A Straight Forward Method: Generate Multiple Sequences

To obtain an estimation at a particular precision level, we need to produce sufficient samples to represent the underlying distribution of interest. This can be obtained by generating a long MCMC chain. However, the longer the chain length, the longer the computation time. To address this problem, sometimes people use multiple processors with each one generating a comparatively short chain, and combine the outputs together for the final estimates.

Although this method may help to improve the simulation efficiency, it also has some problems. For a chain with poor mixing performance and a long burn-in period, generating multiple chains would lead to discarding a larger number of samples in the transient time. Also, we need to consider using different random number seeds assigned to each processor in order to reduce the correlation between samples from different chains.

3.4.2.2 Parallel Metropolis-Coupled MCMC

In classical MCMC algorithms, the proposal distribution is difficult to choose since a large move step can result in a low acceptance probability, while a small step will affect the chain mixing performance and convergence rate [89]. This problem is particularly apparent for a multimodal posterior distribution, where the state might be stuck at one local place, the local optimum, and prohibit better exploration of the entire state-space. However, this is not an insurmountable barrier – an effective method called Metropolis-Coupled MCMC (denoted as $(\mathrm{MC})^3$) was proposed by Geyer in 1991 [90] and first applied for phylogenetic inference [91]. It has been shown to help to improve convergence in [92].

$(\mathrm{MC})^3$ involves multiple chains that are *heated* with increasingly different temperatures to achieve increasingly higher acceptance rates such that a heated chain can move across more modules within a defined sample space, for instance, exploring another peak from a valley in the landscape of the tree in the phylogenetic model. By occasionally swapping the states of the unheated chain (cold chain) with the heated chain, the cold chain is then enabled to jump rapidly from the local optimum, say a deep valley, even within one single step. It is conceivable the cold chain will eventually show better mixing performance. Therefore, with $(\mathrm{MC})^3$, the long chain generated by the classical Metropolis-Hastings algorithm might now be replaced by a short chain such that the whole computation overhead for a cold chain is reduced.

In spite of its benefits, the massive and expensive likelihood calculation resulting from the execution of several chains impairs its practical value. Additionally, more heated chains are probably required for more complex data sets to obtain adequate mixing. Therefore, it is well worth designing a parallel scheme with the aim of cutting down the computation penalty. One parallel $(\mathrm{MC})^3$ algorithm (denoted as P$(\mathrm{MC})^3$) is presented in [93] and its simulation results confirm its ability to achieve nearly optimal speedup for both small and

large data sets.

The straightforward way in a parallel environment is to execute these inherent independent multiple chains on separate processors. However, this might result in two further problems as indicated in [93]: First, for large data structures, the state information might be considerably large in size. In this situation, communication time might even exceed computation time, which easily degrades the speedup. Second, to preserve the limiting distribution as produced in sequential implementation, the parallel version must comply with a strict exchange rule to make sure swaps only take place in the same generations between various chains. The following rule is defined in [93]:

Exchange Rule:
Let $c_{i,k}$ be the i^{th} chain in generation k where $i \in (1, 2, ..., n)$, with n the number of chains, and $k \geq 0$. Chain $c_{i,x}$ can exchange with chain $c_{j,y}$ if and only if $x = y$.

Corresponding suggestions to handle these two difficulties are suggested in [93]: As for the communication cost, it can be reduced by exchanging heat value, rather than state information. Each chain is uniquely characterized by its heat value; once the heats are swapped, these two chains could determine whether or not to accept the newly generated sample following the modified acceptance probabilities associated with their new heat value. Further, originally, there would be two rounds of communication for the swap of both the acceptance probability and the chain states; now only one round is required, that is, if a swap is accepted, the current heat value would be replaced by the new heat value which has already been included in the exchanged swap acceptance information. Hence, using this mechanism, both the number of times and size for message exchanging are decreased. Applying to the phylogenetic problem, different data sets show different performance, among which the highest speedup achieved is about 27 with 32 processors.

Regarding the synchronization requirement determined by the exchange rule, it can be satisfied by the *global* exchange and *point-to-point* exchange schemes. In the global ex-

change scheme, state swapping only happens when every singe chain has carried out the same number of iterations. When they reach the same point, two chains c_i and c_j are randomly selected. During the time when these two chains are exchanging swapping acceptance information, all the remaining chains have to stop to wait until communication is completed. This it to ensure all the chains are at the same generation to adhere to the exchange rule. However, idle time might become the prime cause for a performance penalty. To further increase operation speed, a point-to-point exchange scheme is proposed. Here, c_i and c_j are chosen in advance. And thereby only these two chains need to be in the same generation and the remaining chains can keep on producing the next iteration instead of standing by. Thus, from the perspective of the whole chain generation, simulation time is minimized by getting rid of the wasted time.

3.4.2.3 Parallel Marginalization

Long correlation time could prohibit a Markov chain from rapid convergence to the independent identically distributed samples. With the general recognition in mind that marginal distributions hold shorter correlation time and so faster convergence rate, Weare [94] was inspired by the parallel tempering method introduced in [95, 96] and constructed the parallel marginalization methodology for accelerating MCMC simulation. In this method, several so called *auxiliary chains* with marginal distributions and another chain with the distribution of interest are generated concurrently. By swapping the chain states among auxiliary chains and passing them to the chain sampling from the target distribution, a fraction of parameters which are already in stationary states or at least much more close to the equilibrium state in lower dimensional chains (auxiliary chains) can be brought to higher dimensional chains which perform slower convergence. As pointed out by [94], the strength of this approach is that it would see a guaranteed convergence of the averaging trajectory in terms of preserving the desired distribution.

As described in [94], the first step is to construct L Markov chains with descending dimensions. Let X_0 be the entire parameter space with full dimension d_0 ($X_0 \in \mathbb{R}^{d_0}$) whose probability density is $\pi_0(X_0)$. Its corresponding Markov Chain is $\{x_0^t : t = 0, 1, ..., n\}$ with transition kernel $T_0({x_0}^t \to {x_0}^{t+1})$. For the general case, $\{{x_i}^t, i = 1, ..., L-1\}$ is the lower dimensional Markov chain with limiting distribution $\pi_i(X_i)$, which is a marginal distribution of X_i with dimension d_i, where $d_0 > ... > d_i$. Now factorize X_i as:

$$X_i = (\hat{X}_i, \tilde{X}_i) \tag{3.9}$$

where $\tilde{X}_i$ is to be removed from the current parameter set and $\hat{X}_i$ refers to the remaining variables in the $(i+1)^{th}$ chain of dimension d_{i+1} with $d_i > d_{i+1}$, that is,

$$X_{i+1} \sim \hat{X}_i \tag{3.10}$$

Its corresponding distribution is:

$$\pi_i(X_i) = \pi_i(\hat{X}_i, \tilde{X}_i) \tag{3.11}$$

and it can be further expressed in the form of a conditional distribution:

$$\pi_i(\hat{X}_i, \tilde{X}_i) = \bar{\pi}_i(\hat{X}_i)\pi_i(\tilde{X}_i|\hat{X}_i) \tag{3.12}$$

where $\pi(\tilde{X}_i|\hat{X}_i)$ is the conditional density given $\hat{X}_i$ and $\bar{\pi}_i(\hat{X}_i)$ is the marginal density of $\hat{X}_i$ in the i^{th} chain after removing $\tilde{X}_i$. If calling (3.10), $\pi_{i+1}(\hat{X}_i)$ can be then approximated as:

$$\pi_{i+1}(\hat{X}_i) \approx \bar{\pi}_i(\hat{X}_i) = \int \pi_i(\hat{X}_i, \tilde{X}_i) d\tilde{X}_i \tag{3.13}$$

Now consider the overall statistical properties for the collection of these L independent Markov chains which can be naturally generated in parallel. They are denoted as $\{x^t = (x_0^t, ..., x_L^t)\} \in \mathbb{R}^{d_0} \times ... \times \mathbb{R}^{d_L}$ whose transition probability from x^t at time t to x^{t+1} at

time $t+1$ is given by:

$$T(x^t \to x^{t+1}) = \prod_{i=0}^{L} T_i(x_i^t \to x_i^{t+1}) \tag{3.14}$$

and the equilibrium distribution of $\{X_i : i = 0, ..., L\}$ is:

$$\prod(X_0, ..., X_L) = \pi_0(X_0)...\pi_L(X_L) \tag{3.15}$$

The second step is to exchange the samples between higher and lower dimensional chains such that the states in a rapidly converging trajectory could be eventually passed to the original sample space X_0. Suppose the swap move takes place between the i^{th} and $(i+1)^{th}$ chain. As illustrated in Figure 3.9, the sample $\hat{x}_i^t$ in subset $\hat{X}_i$ from level i with dimension d_i and the sample x_{i+1}^t of variables $\hat{X}_{i+1}$ with dimension d_{i+1}in level $i+1$ (marginal density chain) will be exchanged at time $t+1$. After this, the remaining variables $\tilde{X}_i$ with dimension $d_i - d_{i+1}$ will draw a new sample $\tilde{x}_i^{t+1}$ from the conditional distribution $\pi_i(\tilde{X}_i|X_{i+1})$.

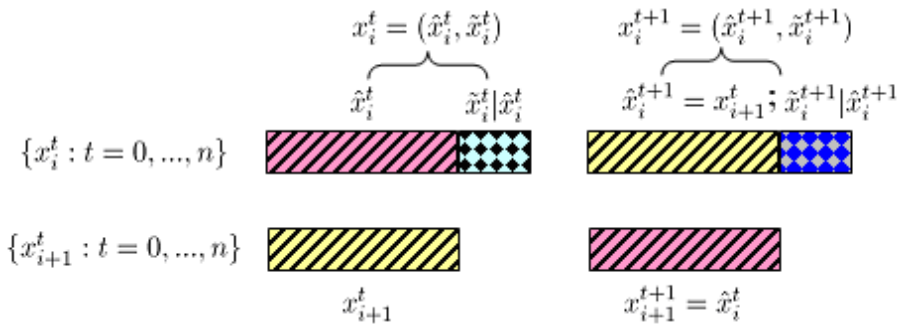


Figure 3.9: Sample swap between $\hat{x}_i^t$ and x_{i+1}^t at time $t+1$. $\tilde{x}_i^{t+1}$ is drawn from $\pi_i(\tilde{X}_i|\hat{X}_i)$ given $\hat{x}_i^{t+1} = x_{i+1}^t$.

To satisfy the detailed balance condition, the above swapping must be accepted with the

swap acceptance probability:

$$A_i = \min \left\{ 1, \frac{\bar{\pi}_i(X_{i+1})\pi_{i+1}(\hat{X}_i))}{\bar{\pi}_i(\hat{X}_i)\pi_{i+1}(X_{i+1})} \right\} \tag{3.16}$$

It can be verified that with the following transition probability $S_i(x^t \to x^{t+1})$:

$$\begin{aligned} S(x^t \to x^{t+1}) = (1 - A_i)\delta_{\{x^{t+1}=x^t\}} \\ + [A_i \pi_i(\tilde{x}_i^{t+1} | \hat{x}_i^{t+1} = x_{i+1}^t)\delta_{\{(\hat{x}_i^{t+1}, x_{i+1}^{t+1})=(x_{i+1}^t, \hat{x}_{i+1}^t)\}} \\ \prod_{j \notin \{i,i+1\}} \delta_{\{x_j^{t+1}=x_j^t\}}] \end{aligned} \tag{3.17}$$

a detailed balance in (3.18) is attained and then the stationary distribution in 3.15 is preserved.

$$\prod(x^t) S_i(x^t \to x^{t+1}) = \prod(x^{t+1}) S_i(x^{t+1} \to x^t) \tag{3.18}$$

Hence, the trajectory average on these L chains will consist to the sample average with respect to the distribution of interest, and then the stationary distribution of original space (d_0) is obtained by averaging over all other auxiliary chains ($d_i : i > 0$) as shown below:

$$\pi_o(X_0) = \int \prod(X_0, ..., X_L) dX_1 ... dX_L \tag{3.19}$$

In the above discussion, both marginal density $\bar{\pi}_i(\centerdot)$ and conditional probability $\pi_i(\tilde{X}_i|\hat{X}_i)$ are supposed to be obtainable for ease of exposition. However, in the general case, they are not always straightforward due to the complex integral computation introduced by marginal distribution calculation. This difficulty could be partially alleviated by clever choice of $\widehat{X}_i$ such that the variable subset is independent of $\widetilde{X}_i$. To completely address this problem, a practical method of approximating marginal density is also proposed by Weare [94]. This method is applied to the bridge sampling problem and the filering and smoothing problem, demonstrating a reduction of autocorrelation from 0.9 to 0.02 and 0.65 to 0.1 repectively.

Despite the improved convergence rate, the effort for marginal density approximation could bring extra computational load. From this point, additional simulation time for marginal density evaluation evaluation might even decelerate parallel running speed instead of producing higher efficiency.

3.4.3 Hybrid Parallel Generation Algorithms

The parallel $(\text{MC})^3$ algorithm presented in the previous section is chain-level parallelization by assigning chains with different heat values onto separate processors. In [89], attempts have been made to realize a hybrid system by exploring an integration of the parallel Felsenstein likelihood [97] calculation method into P(MC)^3. Likelihood computation is implemented on the sub-sequence-level in that the whole data set is divided into segments and each processor is responsible for local likelihood updating for this segment of the entire sequence. Hence, a two-dimensional grid topology is constructed by combining together the chain-level and sub-sequence-level parallelization.

The chains are distributed into r groups and each group is assigned with one chain-level processor. Within each group, the data set for each of the chains is divided into c segments and each sub-sequence-level processor is responsible for computing the local likelihood for one allocated subset. On each row, all the processors should employ the same random number generator RNG_1 with the same random seed such that they could produce the same proposals and perform the same accepting or rejecting action. The intrinsic reason is to ensure the parallel calculation results would consist of the likelihood value evaluated on one processor. However, different rows should have different RNG_1 for randomness diversity. Once local likelihood evaluations are accomplished, they will be gathered together to obtain the global likelihood which will be then broadcasted to all processors on the same row. Thus, two communication rounds take place for one cycle.

According to the swapping scheme in P(MC)^3, strict synchronization is also required in this hybrid approach. To address this issue, another random number generator RNG_2 with a unique random number seed is introduced to all processors to determine which two chains will conduct a possible swap and also decide whether to accept this swap. This mechanism is similar to the point-to-point synchronization technique proposed in P(MC)^3 where only these two randomly chosen chains are requested to be in the same generation and other chains can carry on with their sequence-level proceeding. With the same target of shortening communication message length as in P(MC)^3, temperatures are exchanged rather than chain states.

In addition to synchronization and communication load problems, Fenga *et al.* [89] give further concern to the inherent load balancing difficulty among different rows and proposes two parallel implementation methods. One is called the symmetric parallel MCMC algorithm, where the heat values are exchanged directly between two processors that also participate in sample generation such as the local likelihood calculation here, which is also the method applied by [93]. The other is an asymmetric algorithm where an extra processor is introduced as the coordinate node. Each time a sample generation cycle is complete, the head of each row will send the state information to this coordinate node and when a swap occurs, corresponding processors will fetch it back. As a result, the algorithm can achieve a nearly linear speedup for the specific data set due to the more effective cache usage.

Although this coordinate node could perform other functions, for instance a convergence diagnostic as suggested by [89], it will not anyway take part in sample generation and as a result, it may experience significant idle time. Hence, the hardware device may not be efficiently utilized especially under the case when there are not a large number of chains. Besides this, frequent information gained from chain headers could contribute to extra communication overhead and consequently affect the simulation rate. Regardless of this asymmetric scheme, this hybrid P(MC)^3 algorithm itself could pose a risk to fine-grain

problems due to likelihood parallelization.

3.5 Parallel Likelihood Method for RJMCMC

As pointed out in Section 3.4.1.1, the parallel likelihood method does not destroy the invariance of the generated MCMC chains as well as the sampling scheme. Therefore, it is applicable to the trans-dimensional RJMCMC sampling. Our motivation for RJMCMC analysis has been the interpretation of LaDAR data for 3D imaging. Here, we use the coal mining disaster data in [98] as an exemplar, frequently used for multiple change-point problems, which records the disaster counts between the year 1851 to 1962 (see Figure 3.10). This allows us to compare our results with other published work on this established benchmark, but also has similarities with the more complex problem of LaDAR analysis in that the final data can be considered as a multiplicity of distinct returns (change points or optical reflections).

The observed data $\{y_i : i = 1, 2, ..., n\} \in [0, L]$ is captured by a Poisson process and the Poisson density $x(t)$ is characterized as a step function. The log-likelihood function is expressed as:

$$\sum_{i=1}^{n} \log\{x(y_i)\} - \int_0^L x(t)dt \tag{3.20}$$

Our goal here is to analyze the joint posterior density for $2k + 1$ parameters containing k change-point positions and $k + 1$ Poisson rate in model k, with k the number of change points.

We employ the RJMCMC sampler designed in [39]. The prior information is specified below:

- The number of change point k is drawn from a Poisson distribution: $p(k) = e^{-\lambda}\frac{\lambda^k}{k!}$

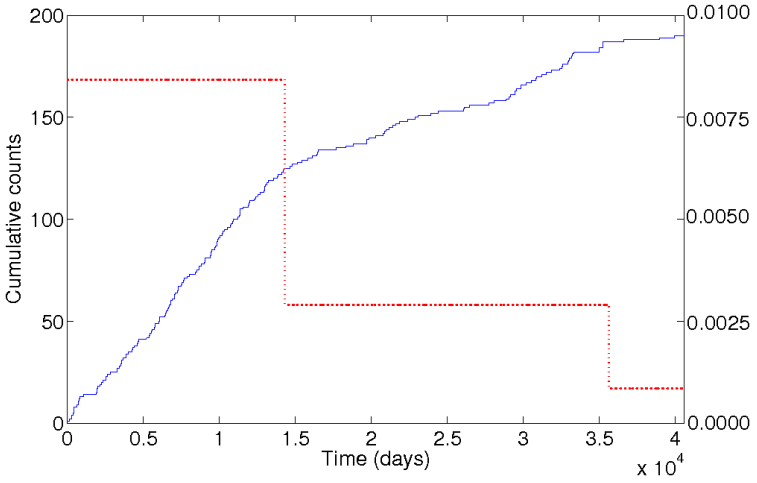


Figure 3.10: Coal mining disaster data, year 1851-1962: cumulative counting process (solid curve) and posterior mode of Poisson rate for $k = 2$ (dotted curve)

- The step changeover point positions $s_1, ..., s_k$ are distributed as even-numbered order statistics from $2k + 1$ points uniformly distributed on $[0, L]$
- The step heights $h_0, ..., h_k$ are independently drawn from a $\Gamma(\alpha, \beta)$ density $\frac{\beta^\alpha h^{\alpha-1} e^{-\beta}}{\Gamma(\alpha)}$ for $h > 0$.

Four move types are defined with their corresponding move probabilities for k change points:

- Position update with probability π_k
- Height update with probability η_k
- Birth of a new step with probability b_k
- Death of an existing step with probability d_k

and satisfying $\pi_k + \eta_k + b_k + d_k = 1$. We run the MCMC sampler for $40,000$ updates with parameter fixed as $\lambda = 3$, $k_{max} = 30$, $\alpha = 1$ and $\beta = 200$. The Figures 3.11 to

3.12 present the estimated posterior model probability and the model specific parameter posteriors.

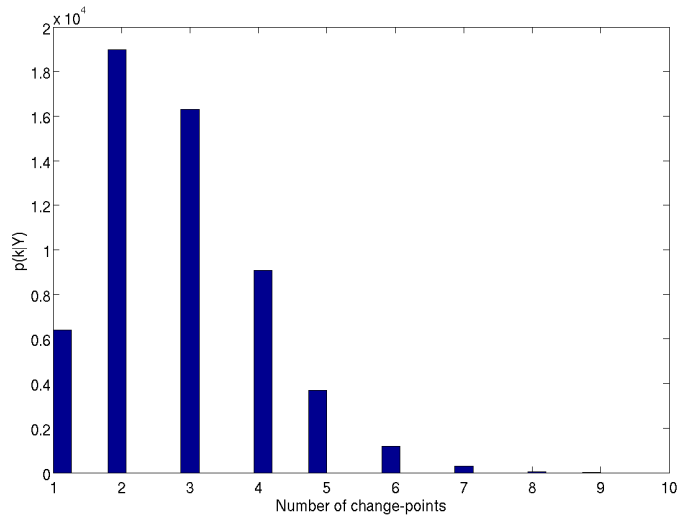


Figure 3.11: Posterior distribution of k, the number of change points.

In RJMCMC sampling, we notice that most of the time is spent for likelihood function calculation. Hence, we distribute the likelihood calculation over a number of processors and allow them to deal with the sub-tasks in parallel. Table 3.1 shows the timing measurements of each individual calculation step in serial calculation of the likelihood function. The first four steps, taking more than 99% of the total execution time, can be implementated in parallel. According to Amdahl's Law, the expected maximum speedup with two processors is 1.99. Table 3.2 presents the parallel procedure and corresponding timing analysis. We observed that the parallel implementation of likelihood calculation even slows down the computation speed for more than 20 times in comparison with the serial implementation. This is caused by the communications overhead, which take about 97% of the total time.

From this example, we learnt that programmers need to be cautious with the parallelization of complex calculation. For some of the cases, although the complex calculations take the majority of computation time in the serial code, it may not be an ideal way to

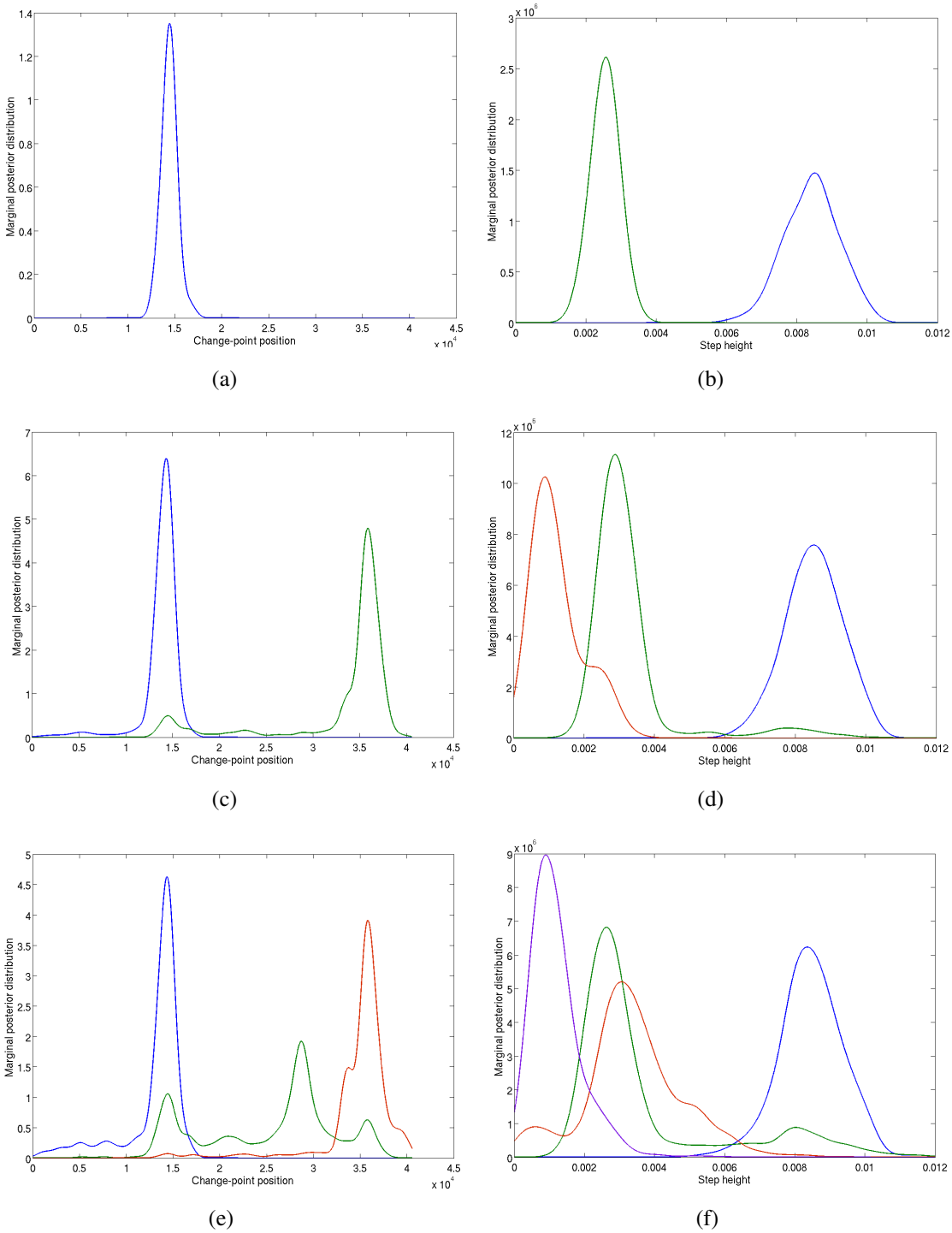


Figure 3.12: Posterior density estimates of change-point positions and step heights conditional on $k = 1$ (a,b), $k = 2$ (c,d) and $k = 3$ (e,f).

Steps	Timing results	%
Get Poisson intensity value for each time point	6.81e−04	39.12%
Get Poisson rate for each disaster	4.18e−04	24.10%
Calculate the summation of Poisson rate	3.84e−04	22.06%
Calculate the product of Poisson rate	9.00e−06	0.52%
Get the final likelihood result	8.00e−06	0.46%
Total time	1.5e−03	100%

Table 3.1: Timing results (sec) for serial likelihood calculation in the coal mining disasters problem.

Steps	Timing results	%
Master broadcasts parameter set to workers	5.6000e−05	0.15%
Master/workers calculate Poisson intensity	3.9300e−04	1.05%
Master/workers calculate the sum of Poisson intensity	1.9500e−04	0.52%
Master collects and calculates the sum of Poisson intensity	2.6700e−04	0.72%
Master copies its partial Poisson intensity	3.6900e−04	0.99%
Master collects partial Poisson intensity in workers	1.4495e−02	38.86%
Master distributes Poisson intensity to workers	1.8860e−02	50.56%
Master/workers calculate Poisson rate	1.2200e−04	0.33%
Master collects and calculate the product of Poisson rate	2.5010e−03	6.97%
Master calculates the final likelihood result	9.0000e−06	0.00024%
Time spent for communication	3.62e−02	97.05%
Total time	3.73e−02	

Table 3.2: Timing results (sec) for parallel likelihood calculation in the coal mining disasters problem using 2 processors. Steps include inter-processor communication are highlighted in bold.

process them in parallel since if the sub-tasks are too fine-grained, communication time may be even larger than sub-task processing time. Therefore, it might be difficult to achieve a speedup.

3.6 A Parallel Algorithm for Trans-dimensional Sampling: Parallel MCMC Chains Method

As discussed in Section 3.3, parallel implementation of MCMC/RJMCMC algorithms in the context of Bayesian inference presents challenges in both the statistical domain and computing domain. For example, the principle factor affecting parallel performance is

the inter-processor communication. When increasing the number of processors, the speed may slow down dramatically due to the additional communications overhead. The parallel computation method illustrated in Section 3.5 provides a good example. Therefore, it is crucial to design an algorithm with relatively low message passing frequency and coarse granularity. Moreover, a particular concern is whether a parallel implementation provides the invariant distribution of interest. Furthermore, since the RJMCMC estimate is based on the trajectory averaging, and the mixing performance can finally determine the mixing performance and convergence length, another significant task is to reduce the correlation among random number streams on separated processors.

In this section, we address the parallel computation for Bayesian model selection with respect to the above considerations. The designed framework takes advantages of the faster convergence rate present in the within-model chains than in a trans-model chain. Speedup is therefore achieved by running shorter MCMC chains in parallel rather than a serial RJMCMC chain.

3.6.1 Parallel MCMC Chains Method

The considerable merit of RJMCMC for across-model simulation is that the joint posterior inference for (k, θ_k) can be obtained directly from a single serial chain. However, to obtain adequate mixing within and between models, a sufficiently long run is required. Our basic premise is that parallel within-model MCMC chains mix better and converge faster than serial RJMCMC chains, and offer coarse grain parallelism that conforms well to the desirable characteristics of parallel MIMD programming. We would expect the within-model chains to have a shorter burn-in period and chain length as each chain only needs to explore one particular parameter subspace and poor mixing between models is also circumvented. Therefore, each within-model MCMC chain proceeds in parallel, with target distribution $p(\theta_k|k, \mathbf{y})$, and the results from separate chains are combined to

construct the posterior inference for k. The within-model posterior density $p(\theta_k|k,\mathbf{y})$ is defined by the parallel MCMC runs, and Bayesian model selection is computed by pairwise comparisons,

$$\frac{p(k_1|\mathbf{y})}{p(k_2|\mathbf{y})} = \frac{p(k_1)}{p(k_2)}\frac{p(\mathbf{y}|k_1)}{p(\mathbf{y}|k_2)} \tag{3.21}$$

where the first ratio on the right-hand side is of prior probabilities, and the second ratio is the *Bayes factor* for model k_1 and k_2 [40]. To estimate the posterior model probability $p(k|\mathbf{y})$, we require the *marginal likelihood* of model k

$$p(\mathbf{y}|k) = \int p(\theta_k,\mathbf{y}|k)\mathrm{d}\theta_k \tag{3.22}$$

As $p(k|\mathbf{y})$ is the normalizing factor of the posterior density, we can express $p(\mathbf{y}|k)$ as:

$$p(\mathbf{y}|k) = \frac{p(\theta_k,\mathbf{y}|k)}{p(\theta_k|\mathbf{y},k)} = \frac{p(\mathbf{y}|k,\theta_k)p(\theta_k|k)}{p(\theta_k|\mathbf{y},k)} \tag{3.23}$$

Equation (3.23) holds for any fixed parameter point of θ_k, say θ_k^*, and now the target becomes the estimation of $p(\theta_k^*|\mathbf{y},k)$ from MCMC runs since we can easily obtain the conditional prior density $p(\theta_k^*|k)$ and the likelihood value $p(\mathbf{y}|k,\theta_k^*)$.

According to [99, 100], although we can choose any θ_k, parameter points with high density are likely to provide more accurate estimation of $p(\theta_k^*|\mathbf{y},k)$. Considering the output of parallel MCMC runs, the θ_k^* corresponding to the posterior mode or maximum likelihood estimate can be selected.

In [99, 100], the $p(\theta_k^*|\mathbf{y},k)$ estimate is a sample average using either the Gibb's sampler or Metropolis-Hastings algorithm. Although these are valid in general, the computation time can grow as the number of mixture components increases since extra MCMC chains are constructed for the marginal posterior estimation. Our alternative method is that when there have been sufficient samples in each parameter subspace as the chains converge to the stationary distribution, we can directly estimate the posterior density $p(\theta_k|\mathbf{y},k)$ using

kernel density estimation techniques.

To summarise, parallel generation of multiple within-model chains has the following advantages. Since the separate MCMC chains are independent, inter-communication is not required and time is not spent in message passing among processors except for initialization and between model selection. Even if computer failures appear, other processors can proceed without interruption; each sequence is still serial, and therefore the desired limiting distribution can be guaranteed. Proper parallelization methods for MCMC chains can also be applied to further speed up single chain generation.

3.6.2 Simulation Results on Coal Mining Disaster Problem

We still use the RJMCMC sampler in Section 2.2.4.2 and restrict the maximum number of change points to be 6 for comparison with [40], since the range $k = 1, ..., 6$ covers most of the posterior probability. To compare the convergence diagnostics and parameter estimations between within-model chains and a trans-model chain, the proposal distributions we choose for MCMC samplers are the same as for the RJMCMC chain but prohibit between-model jumps.

Convergence assessment

In the context of the coal mining disaster problem, the specific target of the RJMCMC sampler is to achieve the Bayesian analysis of both k and the corresponding $2k + 1$ parameters. We aim to carry out the convergence assessment on each individual parameter rather than their scalar summary, in which case the convergence diagnostics defined in [59] are not satisfactory. Therefore, to compare the convergence performance of RJMCMC and MCMC chains, we can also refer to the simulation results in [40], which indicate that the chain has safely converged at 1,000,000 iterations and 200,000 samples could provide very similar posterior density estimation results.

To assess the convergence performance for MCMC chains, we generate four separate sequences for each model, and analyze the diagnostic statistics defined in [48] every 100 iterations after the burn-in period (500 samples). The chain length is set to be 15,000 and Figure 3.13 presents the PSRF values for parameter sets in model $k = 1, ..., 6$. The parameter solution for $k = 2$ is also shown in Figure 3.10. Here, the convergence of a MCMC chain means that the PSRF's for all the parameters reduce to less than 1.2.

It is found that for $k = 1, ..., 6$, within-model chains converge at 350, 5400, 10300, 9900, 9500 and 14400 separately, which means 15000 is a safe length. The reason that the chains with higher dimension might converge slower than lower dimensional chains is that only one single parameter is updated in each iteration and therefore to provide sufficient samples for each dimension, more iterations are required. Unfortunately, this goes against the load balancing requirement for parallel implementation efficiency. This problem could be overcome by updating one set of parameters instead of one parameter in one iteration, or modifying the proposal distribution to improve the acceptance rate to achieve better mixing performance and in turn shorten the chain length as well as the convergence length differences.

Simulation efficiency

To evaluate the parallel improvement, we set the number of iterations for within-model chains to be 15,000 to achieve more accurate estimation of the posterior density $p(\theta_k|\mathbf{y}, k)$, which is in turn used for the Bayes factor calculation, and the length of RJMCMC chain to be 200,000. The timing results are shown in Table 3.3. Using 6 processors in a 32-node Beowulf network, the speed-up of the longest process $(k = 3)$ is 13.04. Although this appears super-linear, the computations are not exactly equivalent. First, the process serial chain performs 2.22 more iterations in total. This would reduce the speed-up to a more plausible 5.87. Moreover, the algorithms are not equivalent, since we cannot predict how much time the serial chain spends in each parameter dimension, so any comparison can only be approximate and may vary from run to run. Nevertheless, there is near-linear

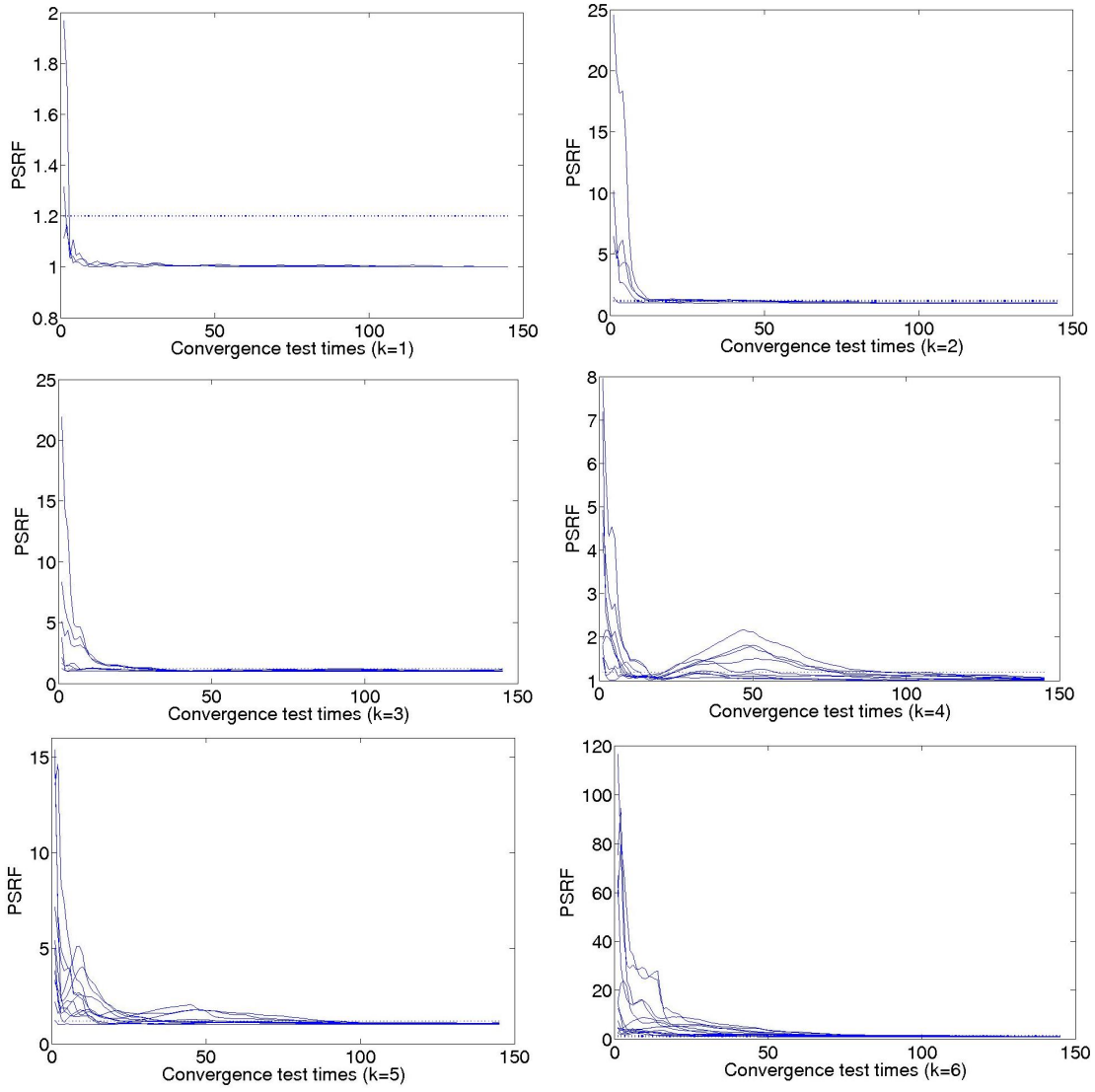


Figure 3.13: Diagnostic plots of $\widehat{R}$ (defined in (2.51)) for within-model MCMC sequences ($k = 1, ..., 6$) of coal mining disaster data for every 100 samples after burn-in period. Dotted curve is the convergence threshold with PSRF=1.2

MCMC	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$
	6.776	6.805	6.926	6.786	6.851	6.759
RJMCMC	90.312					

Table 3.3: Timing results for within-model MCMC chains and trans-model RJMCMC chains measured in seconds.

parallel advantage due to the coarse grain parallelism.

Posterior distribution of model indicator

Figure 3.14 compares the results of $p(k|\mathbf{y})$ obtained from the RJMCMC chain and parallel within-model MCMC chains. First, for the RJMCMC chain, we construct a histogram for k using the direct sampling results and numerically analyze the Bayes factor by selecting samples belonging to different models. This compares these two methods on the same set of samples. The differences between final estimation results probably comes from the approximation of posterior densities in each model using the Gaussian kernel smoother.

Second, we show the calculation results of $p(k|\mathbf{y})$ using samples from RJMCMC chains and multiple MCMC chains. Since they are analyzed in the same way, the results here aim to illustrate the differences of within-model samples from within-model and trans-model chains. The point is that in RJMCMC chains, because of the between model jumps, samples belonging to one particular subspace could be treated as the connection of separate sub-chains. In contrast, MCMC chains provide the continuously serial samples in each model.

3.6.3 Simulation Results on LaDAR Problem

To analyze the performance of the parallel MCMC chains method for LaDAR problem, we employ a set of synthetic data containing five simulated reflection of known, measured instrumental response (Figure 3.15). Here, we incorporate a delayed rejection step with Gaussian proposals for t_0 updates. The standard derivations in the two updating steps are

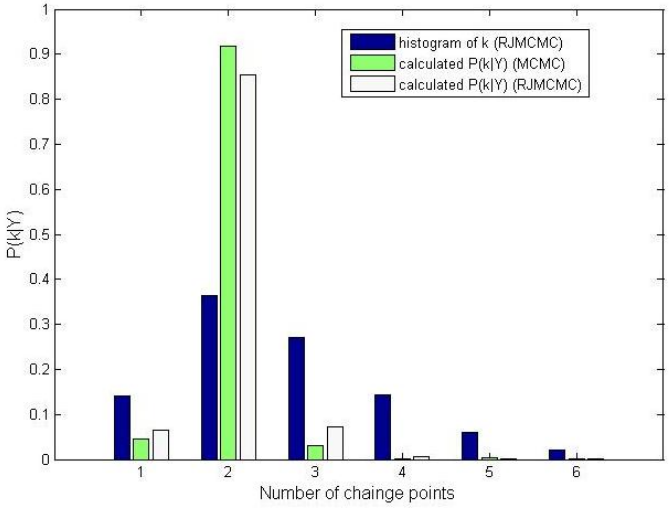


Figure 3.14: $p(k|\mathbf{y})$ obtained from RJMCMC histogram, numerical calculation using RJMCMC samples and numerical calculation using MCMC samples

	Percentage of chains that pass convergence test	Average burn-in length for converged chains	Average chain length for converged chains
k=3	70.0%	996	2791
k=5	5.5%	1504	5819
k=7	0	NA	NA

Table 3.4: Convergence assessment on t_0 for parallel MCMC chains method using synthetic data.

$\sigma_{t_0}^{\text{step1}} = 1000$ and $\sigma_{t_0}^{\text{step2}} = 10$. The first step with a large scale allows a wider exploration range to find the peak, while the small scale proposals in the second step enable exploration around neighbouring values. We anticipate that the delayed rejection step can help to improve the parameter mixing.

Table 3.4 presents the results of convergence assessment from 100 repeat MCMC runs. Figure 3.16 shows the trace plot of positions from random trials. In Figure 3.16(a), we observe that within a very short chain length, less than 500 iterations, both t_0 components jump out of the local refinement of the found peaks and detect the new peaks located in distant regions. Due to the relatively large proposal in the first updating step, the *under-*

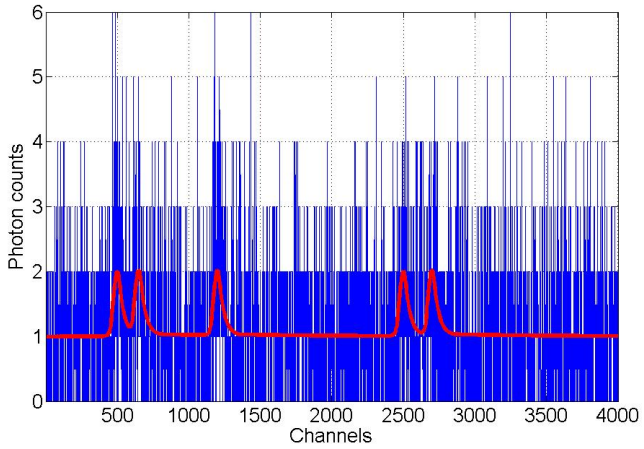
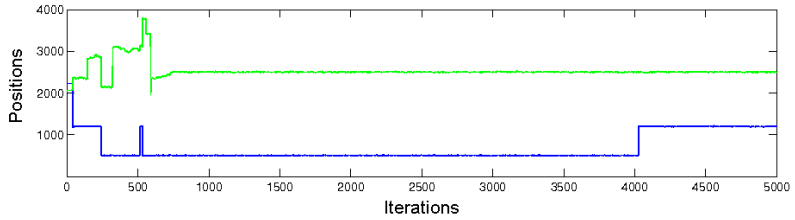


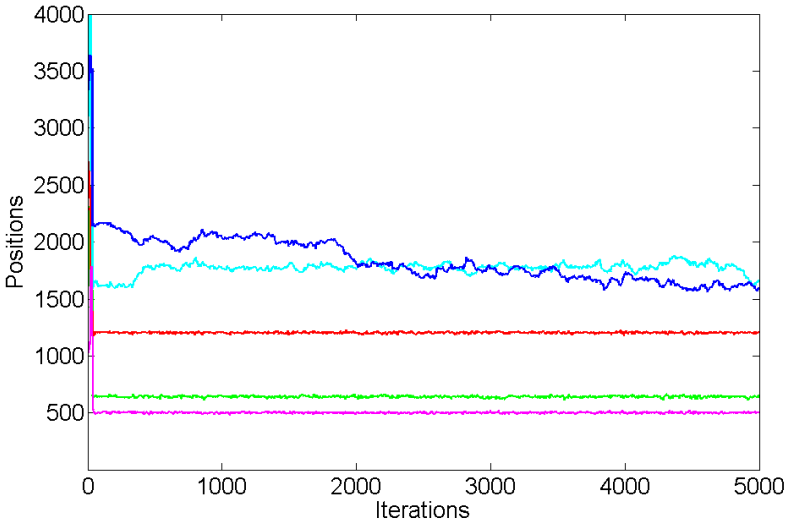
Figure 3.15: Synthetic histogram of photon counts containing five peak returns (blue) and the ground truth (red).

fitting chains with dimensionality less than the ground truth can observe multi-modal marginal posterior density, allowing several high likelihood solutions with very different sets of parameters. On the other hand, in the *over-fitting* chains, the samples will proceed to fit very small false returns and may never achieve a convergence. For this reason, as shown in Table 3.4, none of the 100 MCMC chains can escape from the Markov transient period and eventually pass the convergence test.

Figure 3.16(b) presents the trace plot of MCMC chain for $k = 5$ with random initials. We notice that the delayed rejection algorithm does not function well for all the t_0 components. Only three positions can escape from the transient periods, while the remaining ones are still searching for the proper channel regions after 5000 iterations. This is caused by the conflict requirements of proposal scales in burn-in period and stationary process. At the very beginning, the chain starts with a random initial and all the parameters are within burn-in periods. In this situation, the first step with large proposal scale helps t_0 travel to far away channels and locate the peaks. Thus, more first steps are accepted than the second steps. After a number of iterations, four peaks have found their proper positions, but they would only prefer the small scale proposals to explore the surrounding



(a)



(b)

Figure 3.16: Trace plot of peak positions in MCMC chain with random initials: (a) for $k = 2$, (b) for $k = 5$.

areas and hence reject the first steps. However, positions that have not escaped from the burn-in period still require large proposal scale, but they would not refuse the local updates. In this case, the first step is seldom accepted, and parameters stuck in the transient phase can hardly converge. The conflict of the proposal scales leads to the convergence difficulty of the MCMC chains. As a result, only 5.5% of the sequences can settle down to the stationary state (see Table 3.4).

Since the separate parameter estimates conducted within each model are the preconditions to construct the model posterior probability, the use of parallel MCMC chains method is not valid to interpret LaDAR data.

3.6.4 Discussions and Challenges

We have implemented a method for parallel MCMC chains of different fixed dimension, and compared the results to the prototypical serial RJMCMC chain which allows dimension-changing moves. For the application on coal mining disasters problem, comparing the convergence diagnostics of trans-model and within-model chains, the parallel, multiple within-model chains method improves simulation efficiency because the MCMC chains converge faster than the RJMCMC chains. In a parallel MCMC implementation, proposals are made, accepted or rejected within a fixed dimension, but in serial RJMCMC, the sample space for a given k is explored using steps between different dimensions, which means that the exploration path must be quite different. This warrants further investigation. There is also a difference in the dimensional prediction of the serial RJMCMC and parallel MCMC methods, whether the number of iterations in each dimension or the Bayes factor is used respectively, and these two must be addressed in determining the correct distribution for k. This particular parallel strategy is simple and has obvious benefits, as no communication is necessary during chain generation. In this context, there are many further improvements that could be made, notably in load balancing between chains for differing k since the number of parameter estimations are different in each chain, or in parallelising the individual chains using other methods.

The challenges of the parallel MCMC chains method are as follows: First, it is particularly not valid for LaDAR signal analysis. One possibility to address the convergence problem in MCMC chains is to optimize the sampler updating scheme, such as drawing samples for individual parameters either in a random or sequential order. But this introduces more computation and decreases the sampling efficiency. Second, regardless of the convergence problem in LaDAR, since it is unlikely that all of the chains will converge to the stationary distribution simultaneously with the same length, we need to consider the load-balancing problem. Third, this algorithm is not scalable to the number of processors becaused the number of parallel MCMC chains is fixed and determined by the range of

model dimension.

3.7 Conclusions

In this chapter, a variety of parallel MCMC algorithms are reviewed, commented according to the practical considerations and classified based on whether a single MCMC chain or a group of MCMC chains are split and distributed among a set of processors. We conclude that although MCMC algorithms are serial by nature and not easily migrated onto a parallel system, they can still be processed in parallel by making use of the conditional independence structure of the underlying statistical model.

There have been several strategies for parallel implementation of MCMC algorithms. Although effective in many instances, the vast majority of these strategies are designed specifically for the fixed dimensional MCMC approach, and are not obviously applicable to RJMCMC; hence not feasible for variable dimension problems, such as full waveform LaDAR. For instance, when considering moves between models with different dimensionality, the parameter blocking, parallel Metropolis-coupled MCMC and marginalization algorithms are not valid since the detailed balance requirement for the invariant limiting distribution may not be guaranteed. Some existing parallel MCMC algorithms could potentially be adapted for RJMCMC, but there is a danger of degrading parallel performance by introducing additional undesired calculation or implementation complexity. For example, using the pre-fetching scheme, the desired parallel performance may not be achieved since the between-model moves increase the number of plausible proposals in each updating step. This then requires more processors to calculate the likelihood values simultaneously. Alternatively, using the regeneration algorithm, due to the complexity of the RJMCMC state space and the low re-entry probability to the specific state point, it would be difficult to join together the individual sub-chains to form a single long sequence. In

sum, the development of RJMCMC parallelism remains comparatively neglected.

For the parallel processing of RJMCMC algorithms, we have implemented the parallel likelihood computation method for the benchmark coal mining disaster problem. In the MPI implementation on a Beowulf cluster, it transfers the intermediate results within each sweep in RJMCMC. Because of the fine-grain, the frequent message passing almost sacrifices all the benefits from parallel processing, and we nearly end up with zero speedup.

We have also implemented the parallel MCMC chains method for varying-dimensional problems, which constructed multiple independent within-model MCMC sequences, and combined the separate parameter estimates from individual chains to compute the posterior probability of model dimensions using the Bayes factor. When applied to coal mining disaster data, the parallel MCMC chains mix better and have shorter convergence length than a single RJMCMC chain to explore a mixture of several models. However, there exist convergence problems for LaDAR signals observing multiple surface returns, notably when the model dimension under exploration is not correct, so that multiple incorrect solutions are equally likely.

Hence, it is an urgent task to develop proper parallelization algorithms aiming at analytical tractability for dimension changing problems, specifically the multi-peak detection in LaDAR signals. When designing such an algorithm, we would need to take into consideration the statistical properties of the RJMCMC sampler such as jumps amongst models with different dimensionality. Additionally, another difficulty in the RJMCMC algorithm is the slow acceptance rate of the between-model moves, which will consequently lead to a poor mixing performance and slow convergence rate. Therefore, we will also need to exploit the parallelization from this perspective.

Chapter 4

An Adaptive, Concurrent RJMCMC Method: SSD-RJMCMC Algorithm

RJMCMC is a powerful simulation algorithm to solve Bayesian model determination problems where the parameter vector subject to inference has varying dimensionality. In particular, it has shown considerably better performance than previous methods in resolving closely separated surfaces, and detecting surface returns embedded in high background when analyzing LaDAR data. However, it is computationally expensive since a long chain is usually required for a sufficient between-model mixing. Although several parallel strategies have been proposed to reduce the processing time and memory storage requirements of Markov chain Monte Carlo (MCMC) algorithms, the dimensionality of the parameter vector is known and fixed in that instance. For the parallel algorithms applicable to RJMCMC, such as parallel likelihood computation, we cannot achieve reasonable speedup as expected. Also, the parallel MCMC chains method is not feasible to LaDAR due to the convergence problem in MCMC chains with high dimensionality.

In this chapter, we propose a concurrent, State Space Decomposition RJMCMC (SSD-RJMCMC) algorithm through decomposition and reconfiguration of the state space. It divides the entire set of candidate models (the complete state space) into groups, and

assigns to each an independent RJMCMC sampler with restricted variation of model dimension. Each sampler conducts a local Bayesian model selection. The global result is obtained by comparing and analyzing the local estimates, in conjunction with an embedded error detection and correction scheme for model selection. The parameters for the model of found dimension can be further refined by a following MCMC sampler. The effectiveness of the algorithm is illustrated by application to full waveform LaDAR ranging on both synthetic and real data.

4.1 Motivation for SSD-RJMCMC Algorithm

4.1.1 Discussion on Serial RJMCMC for LaDAR Application

The merit of RJMCMC stands on the trans-dimensional property. First, it enables an automatic comparison between the candidate models and allows the inference of the joint posterior density from a single chain. Second, the between-model jumps can improve the within-model parameter mixing [40]. However, the trans-dimensional jumps also bring weaknesses. First, considering *efficiency*, between-model jumps have lower acceptance rates than fixed dimensional parameter updates, which leads to poor mixing of k and increases the autocorrelation in the realized sequence. For a state space consisting of a large number of models, there may be a long transient time and convergence length. Second, considering *reliability* of model selection, RJMCMC can become stuck in local optima [93], having difficulty in bridging between peaks in a complex posterior distribution. For this reason, even though the diagnostics have indicated convergence, there is a chance that a longer sequence might observe more models beyond the current exploration scope [59].

Applying RJMCMC to LaDAR analysis, the number of surface returns is determined as $\widehat{k} = \text{argmax}[p(k|\mathbf{y})]$, and the sample average of ϕ_k is evaluated using the longest

consecutive segment of the Markov trajectory within the selected model. However, k and ϕ_k may not converge simultaneously. Having identified the convergence of k, there exists a possibility that the limited sample size in the chosen segment may not be large enough to support a full representation of the marginal posterior density $p(\phi_k|\mathbf{y}, k)$, which affects the precision of 3D surface profiling.

4.1.2 Discussion on Parallel MCMC Chains Method for LaDAR Application

As an alternative to sequential RJMCMC, constructing multiple independent within-model MCMC chains has no issues with between-model jumps. First, we can expect better mixing performance and accordingly a lower autocorrelation, which can improve the sampling efficiency and shorten the convergence length. Second, the parallel chains ensure a complete exploration over all the candidate models, which effectively address the local optima problem. Third, without being disturbed by the random jumps, locking the exploration within a particular model guarantees an adequate parameter update, which helps to achieve a high level resolution for LaDAR surface reconstruction.

The ideal case for the parallel MCMC chains is for each sampler of different dimension to have short burn-in, and fast convergence to a true solution. However, it can be difficult for each parameter component to escape from the burn-in period, especially for high dimensional models. Delayed rejection algorithm can help to solve this convergence problem, but as we shown in Section 3.6.3, it is not effective for LaDAR signals involving multiple surface returns.

4.1.3 Motivation

In conclusion, RJMCMC is a computationally intensive algorithm requiring long processing times. This is the motivation for a parallel implementation. Using parallel MCMC chains is a possible solution, but has highlighted problems in highly variable dimensional problems such as LaDAR signal analysis. This leads us to propose the adaptive, concurrent framework, which inherits the benefits of each method and avoids their limitations. We borrow the idea of parallel exploration of the candidate models in the parallel MCMC chains, but retain the trans-dimensional jumps of RJMCMC algorithms.

4.2 SSD-RJMCMC Algorithm

The SSD-RJMCMC framework is divided into four separate stages, as shown in Figure 4.1. Stage 1 implements concurrent RJMCMC sampling, generating a set of independent RJMCMC chains with restricted variation in model dimensionality, and conducts *local* Bayesian model selection. Stage 2 makes *global* model selection, i.e. inference on k. Stages 3 improves the estimates of the within-model parameters. Stage 4 resolves any ambiguities of model selection, providing an error detection and correction scheme to improve reliability.

Figure 4.1: Workflow of SSD-RJMCMC method with four separated stages. LL is the log-likelihood.

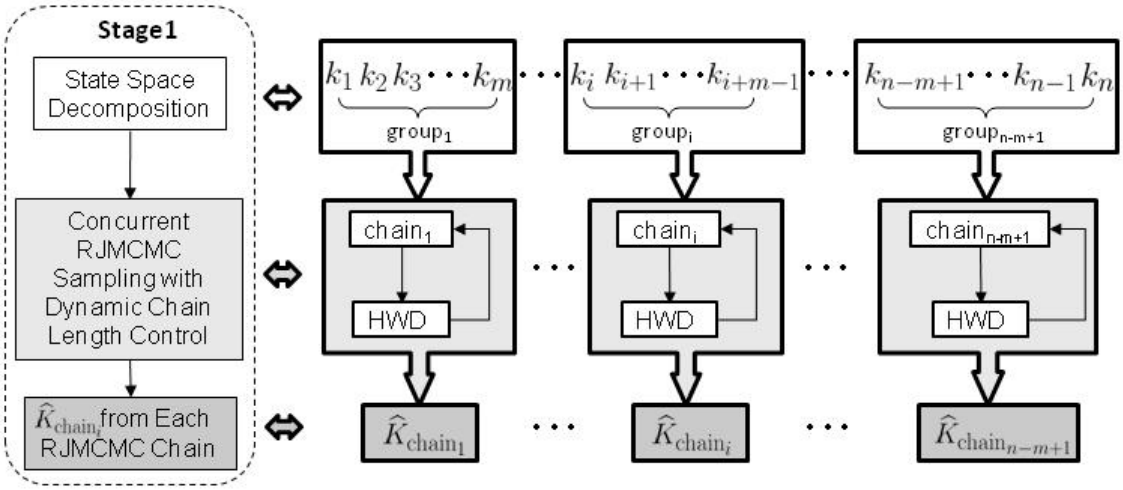


Figure 4.2: Stage 1 in the SSD-RJMCMC algorithm.

4.2.1 Stage 1: State Space Decomposition and Local Model Selection

For concurrent sampling in an RJMCMC framework, the complete state space of n candidate models, $\{k_1, k_2, ..., k_n\}$ is divided into $(n - m + 1)$ groups, each containing m adjacent models, i.e. group$_i$ $= (k_i, k_{i+1}, ..., k_{i+m-1})$, as illustrated in Figure 4.2. Each group is assigned an independent RJMCMC chain to explore the restricted models, hence reducing the between-model mixing complexity by reconfiguring the state space.

Specifically, we set m to be 3, i.e. group$_i$ $= (k_i, k_{i+1}, k_{i+2})$. This produces a set of triple-state RJMCMC chains, wherein the samples can jump from the centred model to models either with higher dimension through Birth/Split or lower dimension through Death/Merge operations. Moreover, Since the neighbouring groups possess two adjacent models in common, all the models can become centred one except for k_1 and k_n. This guarantees complete exploration of the entire state space. Furthermore, because the neighbouring chains provide reinforcement for pairwise comparison of k, the reliability of model selection is enhanced, particularly in the circumstance of poor mixing performance. This benefit will be lost if we set $m = 2$, since each pair of the adjacent models

will only be compared once. Finally, increasing the value of m, for instance, $m = 4$, adds in candidate models in each group. In this case, we cannot reduce the between-model mixing difficulty to the largest extend and benefit the most from the space decomposition.

In each triple-state RJMCMC sequence, we apply Heidelberger and Welch diagnostic (HWD) to assess convergence and dynamically terminate the chain generation. For the converged chain, we conduct the *local* Bayesian model selection and obtain $\{\widehat{K}_{\text{chain}_i} : i = 1, 2, ..., n-2\}$ corresponding to the highest marginal posterior probability $p(k|\mathbf{y})$ in each group.

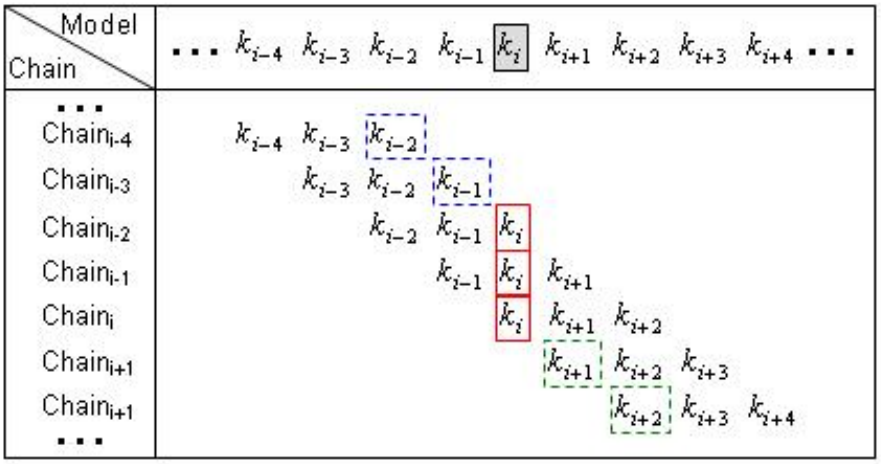
4.2.2 Stage 2: Global Model Determination

Based on $\{\widehat{K}_{\text{chain}_i}, i = 1, 2, ..., n-2\}$ from the parallel triple-state RJMCMC chains in Stage 1, we make a *global* model determination by comparing the *local* results. Generally, there are two cases:

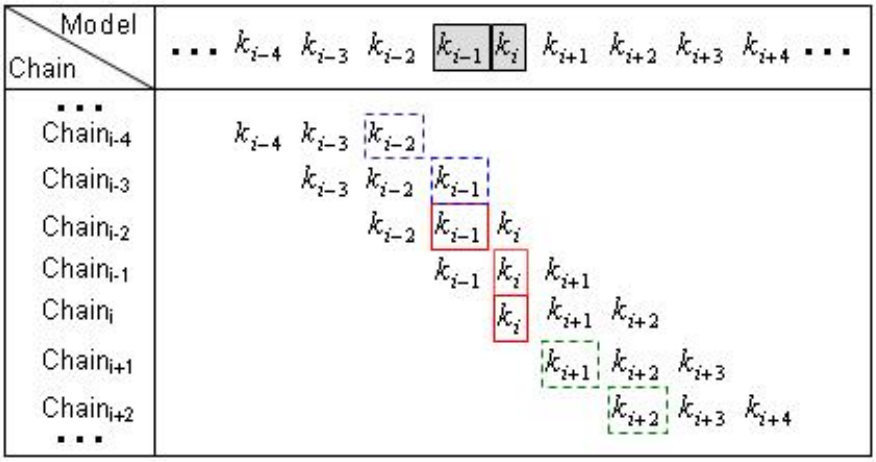
- Case 1: *Single solution for global model determination.* As displayed in Figure 4.3(a), if all triple-state RJMCMC chains pick up the correct model, that is, the under-fitting chains select the upper bound, over-fitting chains select the lower bound, and the remaining chains give the ground truth, there should be three adjacent parallel chains concluding the same k_i. Accordingly, the final model selection result is $\widehat{K}_{\text{final}} = \widehat{K}_{\text{global}} = k_i$.

  An exception to the rule occurs at the extrema, k_1 or k_n. If so, all the triple-state chains should choose the lower or upper bound, and the corresponding solution is the minimum or maximum value of k.

- Case 2: *Multiple possible solutions for global model determination.* Suppose one



(a)



(b)

Figure 4.3: Stage 2 in SSD-RJMCMC algorithm. (a) Case 1: single solution (grey-filled square) for global model determination, $\widehat{K}_{\text{global}} = k_i$, with illustrations of local model selection results for under-fitting chains (dashed blue), over-fitting chains (dashed green) and the chains containing the true model (solid red). (b) An example of Case 2, multiple possible solutions for global model determination, $\widehat{K}_{\text{global}_1} = k_{i-1}$ and $\widehat{K}_{\text{global}_2} = k_i$.

of the chains that contains the ground truth makes an error in local model selection, for instance, chain_{i-2} gives $\widehat{K}_{\text{chain}_{i-2}} = k_{i-1}$ in Figure 4.3(b). In this situation, we observe k_{i-1} and k_i twice in the local results, and treat both as the possible global solutions in succeeding stages, i.e. $\widehat{K}_{\text{global}_1=k_{i-1}}$ and $\widehat{K}_{\text{global}_2=k_i}$.

4.2.3 Stage 3: Parameter Estimates for Chosen Model(s)

Although k has converged, within-model mixing in RJMCMC chains may not be adequate to fully represent the parameter subspace of the model-specific parameter posterior $\pi(\phi_{\widehat{K}_{\text{global}_j}}|\mathbf{y}, \widehat{K}_{\text{global}_j})$. A subsequent MCMC chain, or chains, adds sufficient within-model exploration and achieves a high level accuracy of parameter estimates. However, this succeeding stage is optional depending on convergence assessment for the parameter vectors in Stage 1 as shown in Figure 4.1.

First, for Stage 1 chains, we extract the longest consecutive segment for the selected model within and between chains. Second, for sample trajectories in the longest segment, we apply the HWD convergence diagnostic. If the parameter components pass the convergence test, there is no need for further processing. Otherwise, we extend the segment using its last sample to initialize an MCMC process and continue within-model updates. We then monitor convergence for the concatenated RJMCMC segment and the newly generated samples.

4.2.4 Stage 4: Optional Between-model Comparison

This occurs only when there is more than one possibility for global model determination (Case 2) in Stage 2. Knowing the within-model parameter estimates for each possible model $\widehat{\phi}_{\widehat{K}_{\text{global}_j}}$ in Stage 3, the model with the largest likelihood value is the final solution,

i.e. $(\widehat{K}_{\text{final}}, \widehat{\phi}_{\widehat{K}_{\text{final}}}) = \text{argmax}[\widehat{LL}_{\widehat{K}_{\text{global}_j}}]$.

4.2.5 Convergence Assessment in SSD-RJMCMC

To control chain generation, we assess convergence on both k and ϕ_k using HWD. To achieve the same convergence level for all the parameter components regardless of their values, we replace RHW (defined in (2.58)) with the absolute half-width (AHW),

$$AHW = z_{1-\alpha/2}\sqrt{\widehat{S}(0)/n_p}. \tag{4.1}$$

For a gradual monitoring of the convergence performance, we apply the diagnostic every ΔN samples after generating a minimum number of iterations N_0, rather than using the originally defined intervals with increasing length.

For the MCMC chains, we apply HWD to t_0 and stop sampling on convergence, as the priority of LaDAR is to estimate the range associated with the peak positions. For the RJMCMC chains, either standard or the triple-state chains in SSD-RJMCMC, we assess the convergence on k, and stop the chain generation when it passes the AHW test. The significance level α is set as 0.05. N_0, ΔN and the AHW thresholds are specified within the different experiments.

4.3 An Illustration of SSD-RJMCMC Method on Synthetic Data

Our initial evaluation uses a challenging, synthetic LaDAR response illustrated in Figure 4.4. The first advantage of using synthetic data is that it represents a difficult detection and resolution problem as we set the signal-to-background ratio (SBR) of the five, closely

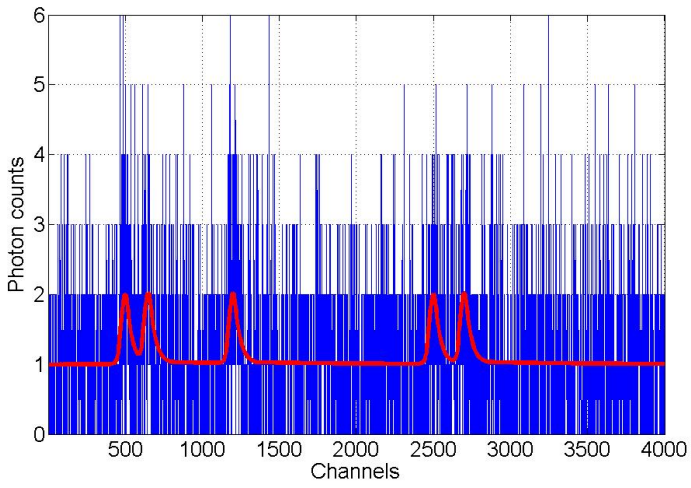


Figure 4.4: Synthetic histogram of photon counts containing five peak returns (blue) and the ground truth (red). The specified locations of isolated peak returns are $t_0 = \{500, 650, 1200, 2500, 2700\}$. Peak amplitudes and background level are set to 1.

spaced returns to 1. Further, we know the absolute ground truth for evaluation. In testing, we constrained the possible number of peaks from 0 to 9, giving eight triple-state chains (chain_0 to chain_7), and the starting values of k in the reversible jump chains were random. N_0 and ΔN in HWD were chosen to be 1000 and 500 respectively. The previously unspecified AHW thresholds for t_0 and k were set as $\text{AHW}_{t_0} = 5$ and $\text{AHW}_k = 0.15$. In this example, we have deliberately made the the convergence condition of k loose in Stage 1 of the algorithm, which makes model selection error more likely from inadequate between-model mixing, but does lead to faster convergence. However, these potential errors can be corrected by the subsequent stages and hence reduce the overall complexity, as we investigate more fully in Section 4.4.

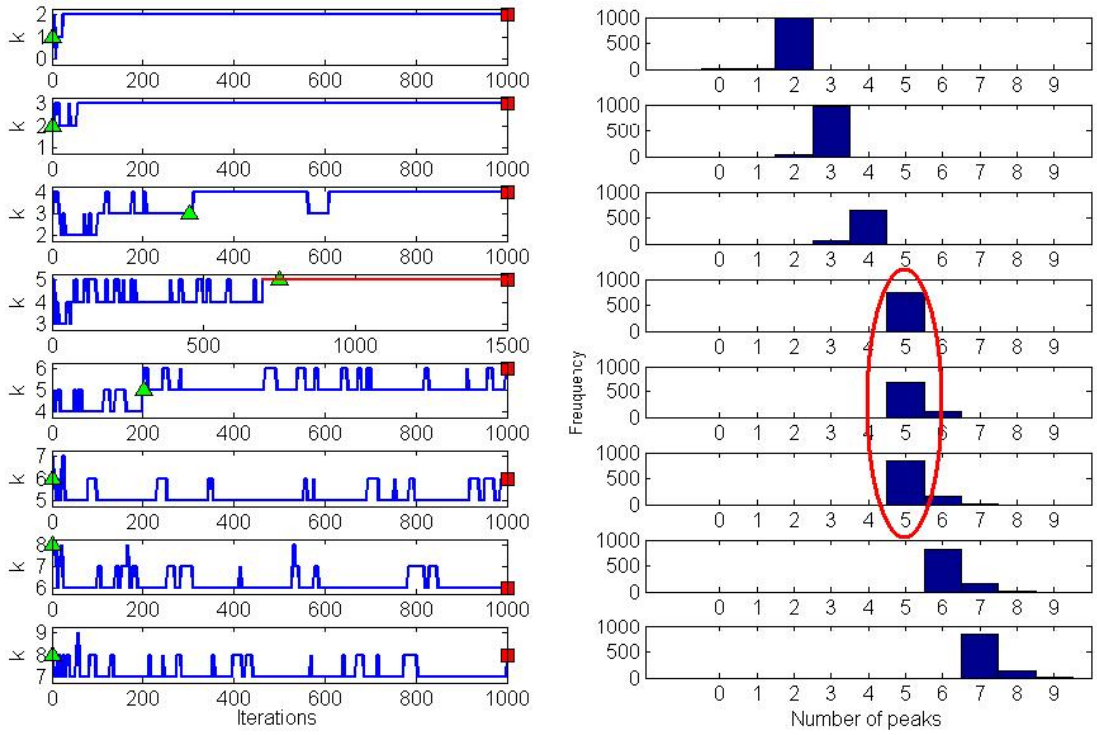
4.3.1 SSD-RJMCMC: Single Solution for Global Model Determination

Stage 1

Sample trajectories for k with marked burn-in periods and convergence lengths are shown in Figure 4.5(a). For under-fitting chains, $\{\text{chain}_0, \text{chain}_1, \text{chain}_2\}$, there is rapid convergence to the upper limit. After reaching the covariance stationary states, k is almost constant and has extremely small variance. Consequently, the sequence achieves the AHW bound within the first convergence check point N_0. The distinct histograms of $P_{\text{chain}_j}(k|\mathbf{y})$ in Figure 4.5(b) show that local Bayesian model selection results are at the higher limits of model dimension.

The three adjacent chains containing the true model have different mixing performance. For example, in chain_3 ($k = 3, 4, 5$), there are frequent moves between $k = 4$ and $k = 5$ during the burn-in period, but afterwards the chain is fixed at the correct value. By contrast, in chain_4 and chain_5, there are still frequent moves between $k = 5$ and $k = 6$ after burn-in, since the prior probability for peak amplitude in between-model jumps encourages the detection of small and inconspicuous peaks (see Section 2.2.2.2). These continual jumps increase the sample variance, but because of the reduced size of the state space in comparison with serial RJMCMC, convergence can still be achieved with relatively short chains, 1500 and 1000 respectively in this random trial. As expected, the local model selection results for these chains are all the same, $\widehat{K}_{\text{chain}_j} = 5$ for $j = 3, 4, 5$.

The over-fitting chains, chain_6 and chain_7, continue to fit the small false returns. As a consequence, parameters associated with the model dimensions, such as t_0 and β, may never converge to a consistent solution. Nevertheless, this does not prevent convergence in k as the sampler can still compare between models with different dimensionality. Accordingly, the histograms favour the models possessing fewer peaks.



(a) (b)

Figure 4.5: Stage 1 and Stage 2 in the adaptive concurrent method in the case of single $\widehat{K}_{\text{global}}$. (a) demonstrates the state space decomposition and sample trajectories for k in all the triple-state RJMCMC chains with marked burn-in periods (green triangle) and stop lengths (red square) determined by HWD. (b) presents the histograms of k for local model selection.

Stage 2

Figure 4.5(b) gives a single solution for global model determination, i.e. $\widehat{K}_{\text{final}} = 5$.

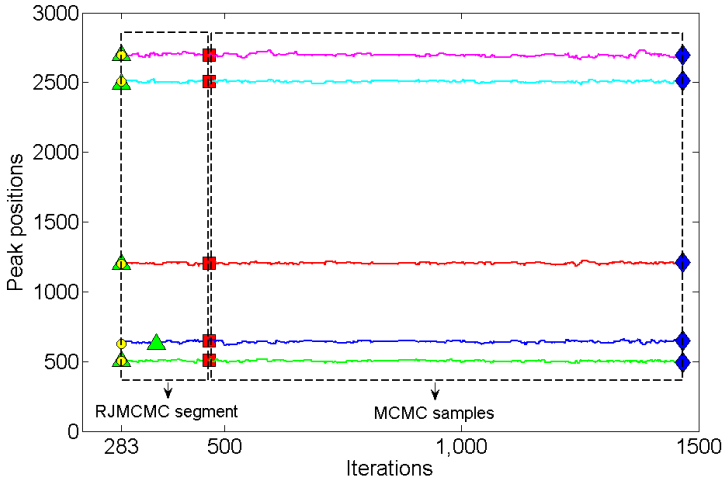
Stage 3

Using the already formed chain$_4$, we analyze the longest consecutive segment with $k = 5$. All positional estimates have already converged except for t_{0_j} around 2700. Therefore, a MCMC chain, initialized with the last sample in this segment, is generated and all positions have converged after 1000 more updates. To give a comparison for parameter subspace explorations, we plot the histogram of $p(t_{0_j}|k, \mathbf{y})$ for the RJMCMC segment without (Figure 4.7(a) to (e)) and with the extra MCMC samples (Figure 4.7(f) to (j)).

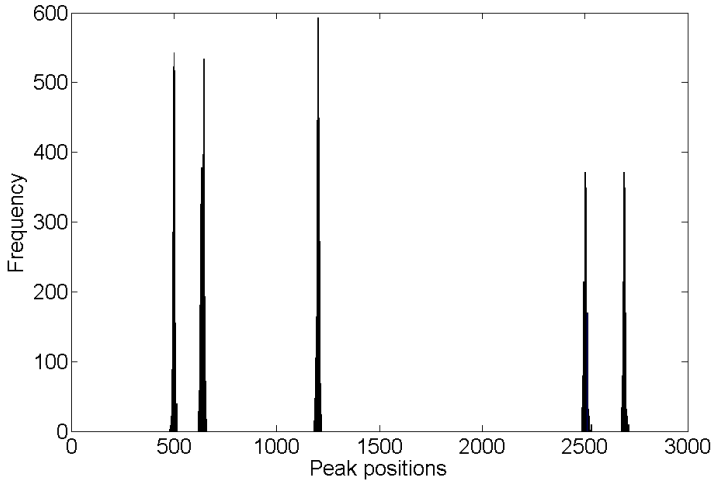
	Actual values	Without MCMC	With MCMC
t_{0_1}	500	498.7	501.2
t_{0_2}	650	640.6	641.6
t_{0_3}	1200	1203.9	1202.9
t_{0_4}	2500	2510.9	2507.8
t_{0_5}	2700	2690.7	2694.7
β_1	1	1.35	1.20
β_2	1	1.03	0.95
β_3	1	1.36	1.27
β_4	1	1.32	1.05
β_5	1	0.83	0.67
B	1	1.03	1.01

Table 4.1: Parameter estimates from the highlighted consecutive segment for $\widehat{K}_{\text{final}} = 5$ in Figure 4.5(a), and with the additional MCMC chain (1000 more samples for this trial) in Stage 3 of the SSD-RJMCMC approach.

Table 4.1 demonstrates that the following MCMC sampler can improve the estimation accuracy of t_0 by providing more adequate within-model parameter exploration, showing the occasional necessity of this complementary stage.



(a)



(b)

Figure 4.6: Stage 3 in SSD-RJMCMC in the case of single $\widehat{K}_{\text{global}}$. (a) Trace plot of positions in RJMCMC segment from chain κ_{46} in Figure 4.5(a) highlighted in red, and continuing MCMC chain. Green triangles and red squares are burn-in and convergence length for the RJMCMC segment, yellow balls and blue diamonds are burn-in and convergence length for both of the segment and newly generated MCMC samples. (b) Histogram of all peak positions after convergence.

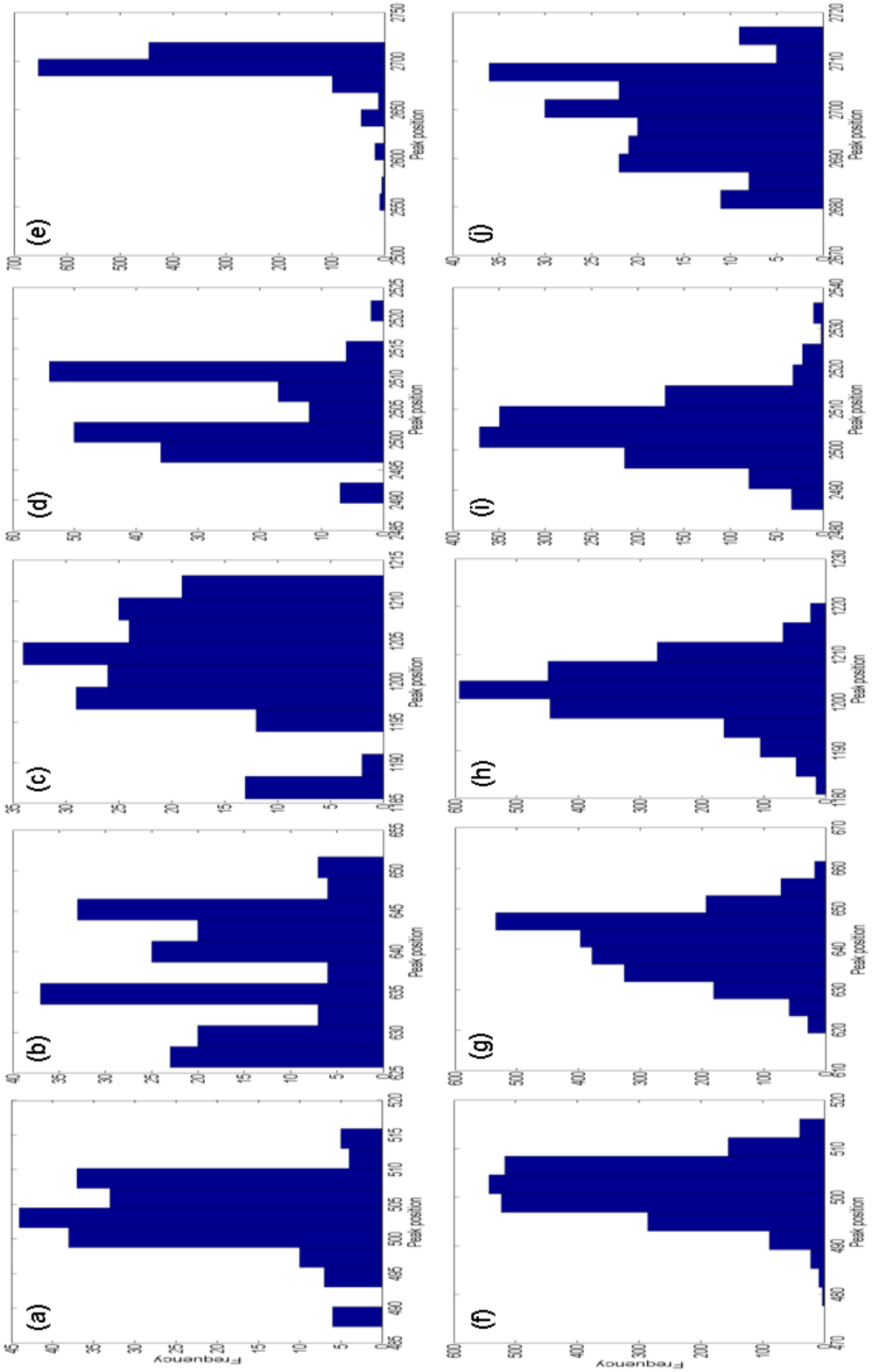


Figure 4.7: (a) to (e) are histograms of $p(t_{0_j}|k, \mathbf{y})$ for RJMCMC segments (hilighted with red colour in Figure 4.5(a)) with additional MCMC samples shown in Figure 4.6(a). (f) to (j) are histograms of $p(t_{0_j}|k, \mathbf{y})$ for the same RJMCMC segments but without additonal MCMC samples.

4.3.2 SSD-RJMCMC: Multiple Possible Solutions for Global Model Determination

Stage 1

We show an alternative example in Figures 4.8(a). The key difference is that chain_3 accepts many jumps throughout the sequence and gives $\widehat{K}_{\text{chain}_3} = 4$, rather than 5. We notice that this chain behaves in a similar manner as the transient period of chain_3 in Figure 4.5(a), which indicates insufficient between-model mixing and a local convergence.

Stage 2

Recalling Figure 4.3(b), we conclude two possibilities for global model selection, $\widehat{K}_{\text{global}_1} = 4$ and $\widehat{K}_{\text{global}_2} = 5$, and pass both of them to Stage 3 for parameter extraction. Unlike a single standard RJMCMC chain which does not determine the local convergence, the concurrent structure with overlapped models enables us to detect the local model selection error.

Stage 3

Convergence assessment shows that not every component of t_0 has converged. Therefore, to obtain the desired parameter estimation accuracy, we assign both models an independent MCMC chain, which converge after 900 and 300 samples for $\widehat{K}_{\text{global}_1} = 4$ and $\widehat{K}_{\text{global}_2} = 5$ respectively. Corresponding sample trajectories are shown in Figure 4.8(c) and (d).

Stage 4

By substituting the extracted parameter estimates in Equation (2.4), we obtained the log-likelihood values, -3.899e+003 for $\widehat{K}_{\text{global}_1} = 4$ and -3.886e+003 for $\widehat{K}_{\text{global}_2} = 5$, which gives $\widehat{K}_{\text{final}} = 5$ and the associated within-model parameter vector as a final result.

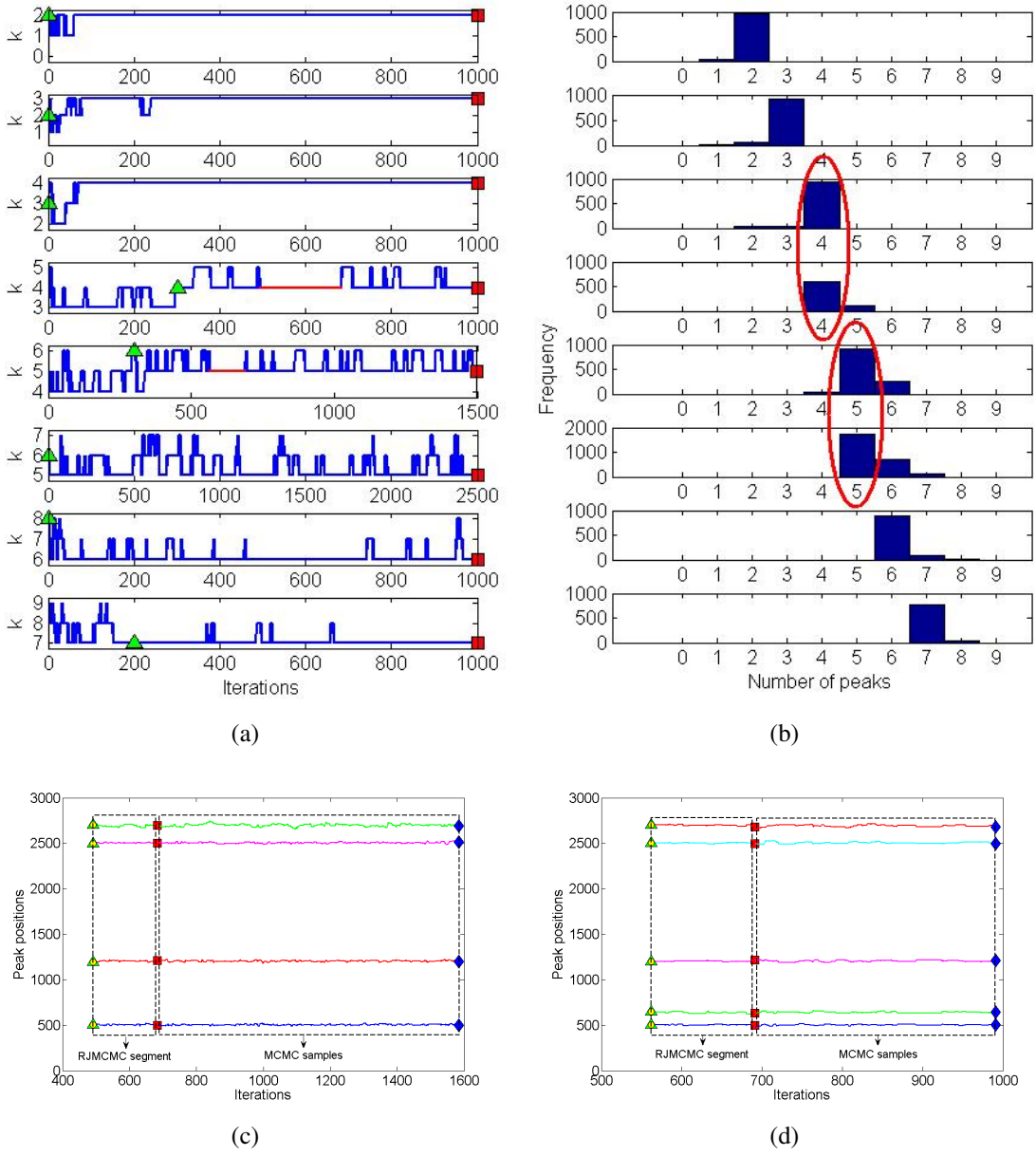


Figure 4.8: Stage 1 and Stage 2 in the adaptive concurrent method in the case of multiple $\widehat{K}_{\text{global}}$. (a) Stage 1: state space decomposition and sample trajectories for k in all the triple-state RJMCMC chains with marked burn-in periods (green triangle) and stop lengths (red square) determined by HWD. (b) Stage 2: histograms of k for local model selection. (c) Stage 3: Traceplot of selected RJMCMC segment and continuing MCMC samples for $\widehat{K}_{\text{global}_1} = 4$. (d) Stage 3: Traceplot of selected RJMCMC segment and continuing MCMC samples for $\widehat{K}_{\text{global}_2} = 5$.

4.3.3 Serial RJMCMC Algorithms with Comparison to SSD-RJMCMC

Figure 4.9 gives a set of sample trajectories of k and the corresponding histogram $p(k|\mathbf{y})$. In the first plot (Figure 4.9(a)), the chain converges at 6000 iterations with a initial transient period of 600. We observed that the chain needs to travel through more models compared with triple-state RJMCMC before escaping from the burn-in period. After that, samples jumps among models from $k = 4$ to $k = 8$, which hold higher posterior probability and are the underlying steady states of the realized chain. This verifies our discussion in Section 4.1.1 that standard RJMCMC have larger state space and hence the sample divergence, which results in a longer convergence length. For this reason, concurrent RJMCMC chains are more efficient for model selection.

To evaluate the accuracy of parameter estimates, we select the longest consecutive RJMCMC segments with $k = 5$ (iteration 2993 to 3334) and apply HWD on t_0. The convergence assessment shows that t_{0_j} around 650 and 2700 fail the AHW test, and burn-in exists in the former one. Histograms of $p(t_0|\mathbf{y}, k = 5)$ are provided in Figure 4.10, which discloses a less adequate within-model exploration in comparison to Figure 4.7. This implies that k and θ_k may not converge simultaneously, that is, passing the AHW test on k does not naturally guarantee the convergence on t_0.

The remaining trails in Figure 4.9 demonstrate the reliability problems with RJMCMC discussed in Section 4.1.1. The second run displayed in Figure 4.9(c) converges at 1000 iterations, and the histogram of k shown in Figure 4.9(d) gives $k = 4$. To explain the model selection failure, we refer back to the synthetic data set. The high value of SBR and closely separated peaks yield considerable challenges for signal interpretation. As a consequence, chains would have difficulty to resolve all the peaks correctly using a short chain, because it might have not sufficiently explored the entire model space, even though the diagnostic indicates a converge. This can also happen in concurrent RJMCMC chains of SSD-RJMCMC. However, with the reconfigured state space and the overlapped

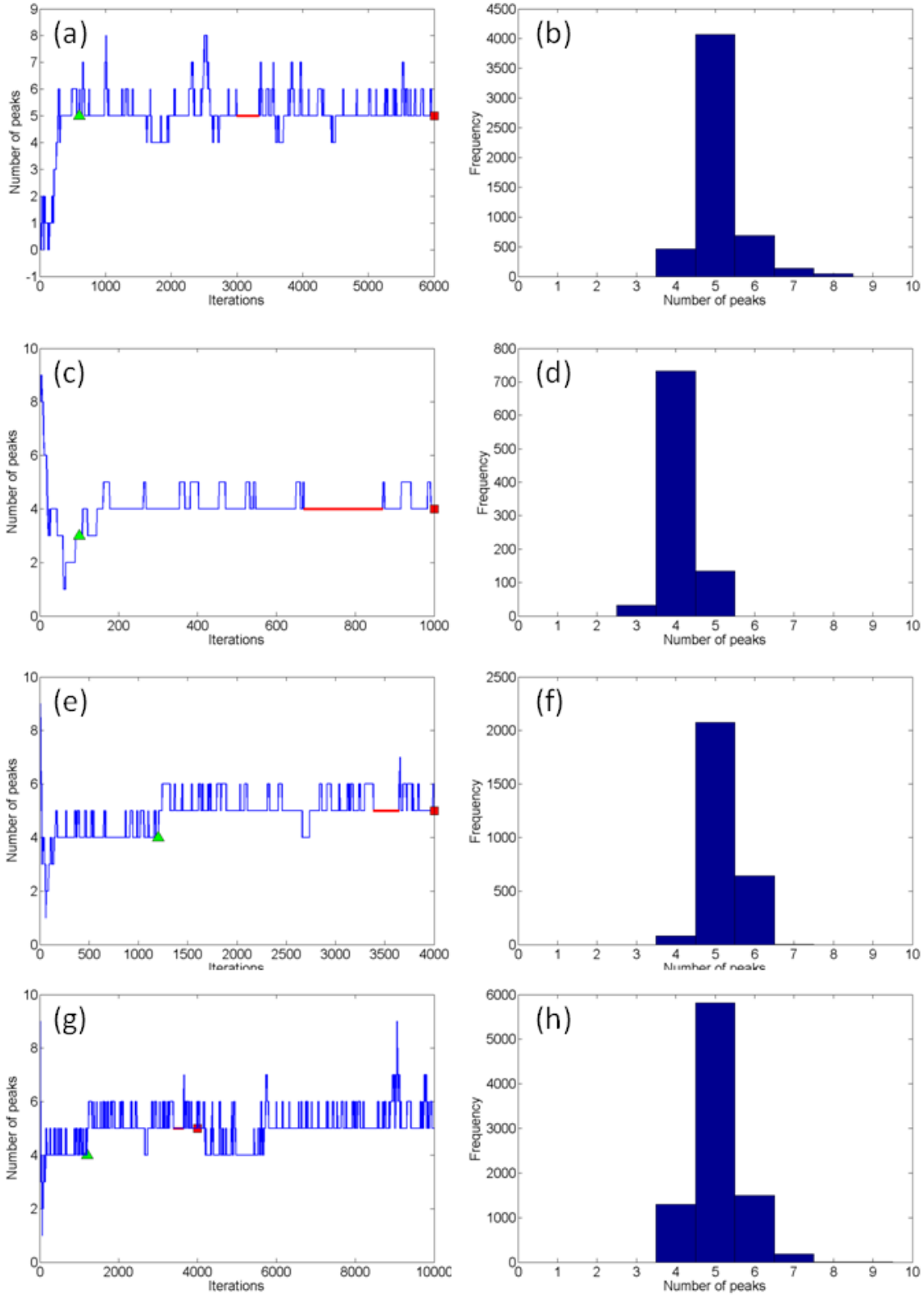
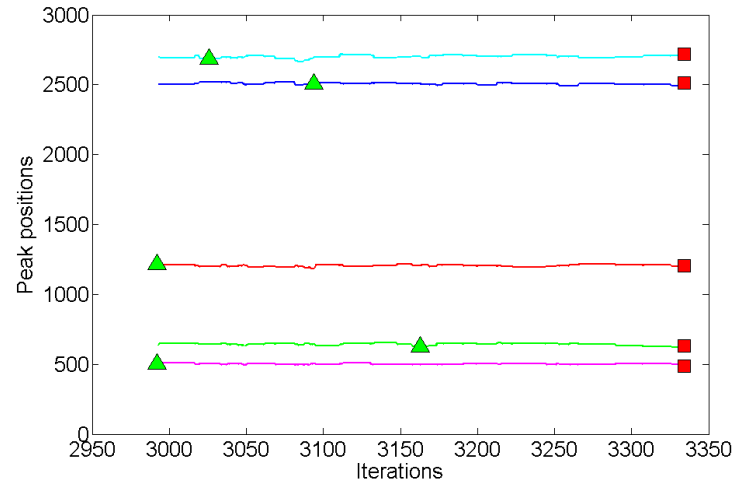
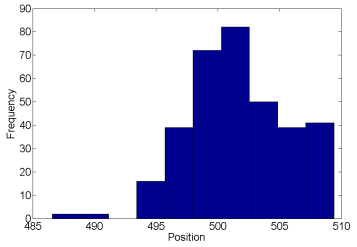


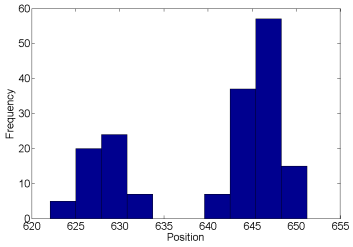
Figure 4.9: Single run using RJMCMC with Heidelberger and Welch convergence diagnostic.



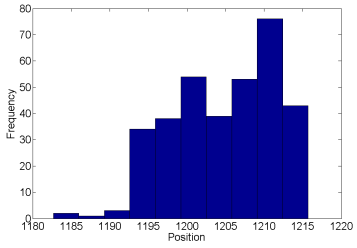
(a)



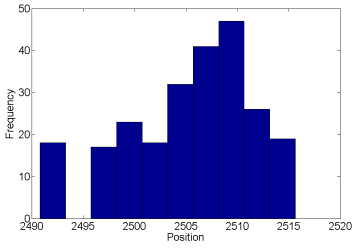
(b)



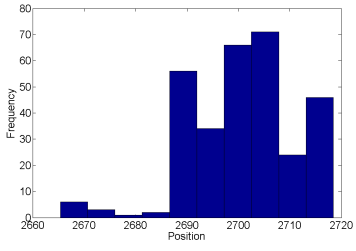
(c)



(d)



(e)



(f)

Figure 4.10: Convergence assessment on peak positions in RJMCMC segment and MCMC chain with random initials. (a) Traceplot of peak positions in the RJMCMC segment (highlighted with bold red in Figure 4.9(a)) (b) to (f) histogram from traceplot of positions in (a). (b) for megenta, (c) for green, (d) for red, (e) for blue, (f) for cyan. (c) fails the convergence test, while others pass the convergence test.

models in adjacent triple-state chains, SSD-RJMCMC can identify the local convergence and recognize the local Bayesian model selection error.

One possible solution to address the local convergence problem in serial RJMCMC is to postpone the first convergence monitoring point. If we continue the sequence and extend N_0 to 1500, we notice that the first 1200 iterations are detected as burn-in period and the chain length rises from 1000 to 4000 as shown in Figure 4.9(e). This suggests that to avoid a local convergence, N_0 needs to be sufficiently large.

Although the RJMCMC chain shown in Figure 4.9(e) gives the correct answer for model selection, it does not observe an adequate between-model mixing. In comparison with the histogram of $p(k|\mathbf{y})$ presented in Figure 4.9(b), the dominant models in Figure 4.9(f) are $k = 4$ to $k = 6$. If we draw more samples (see Figure 4.9(g)), model $k = 8$ is then included within its exploration scope.

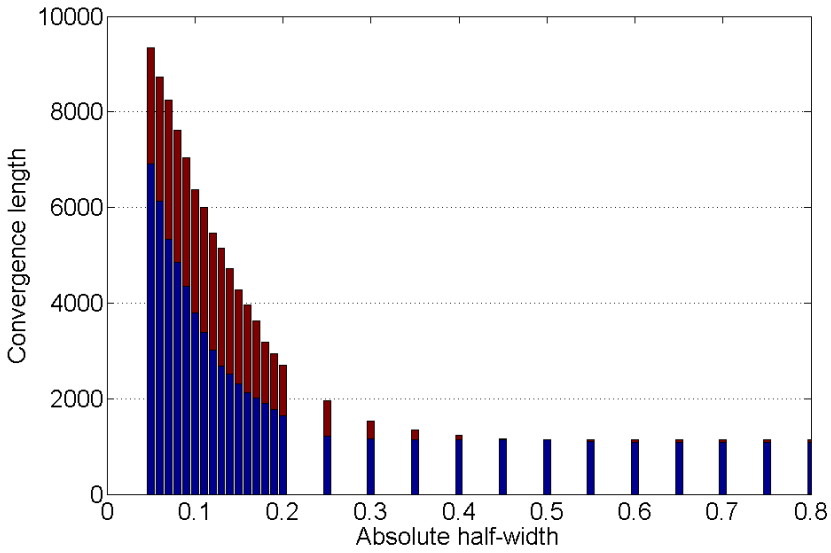


Figure 4.11: Average convergence length of the longest chain in SSD-RJMCMC (blue bar) and standard RJMCMC (red bar) methods for the synthetic data using the same AHW_k values.

Figure 4.11 compares the average convergence length (over on 100 random trials) of

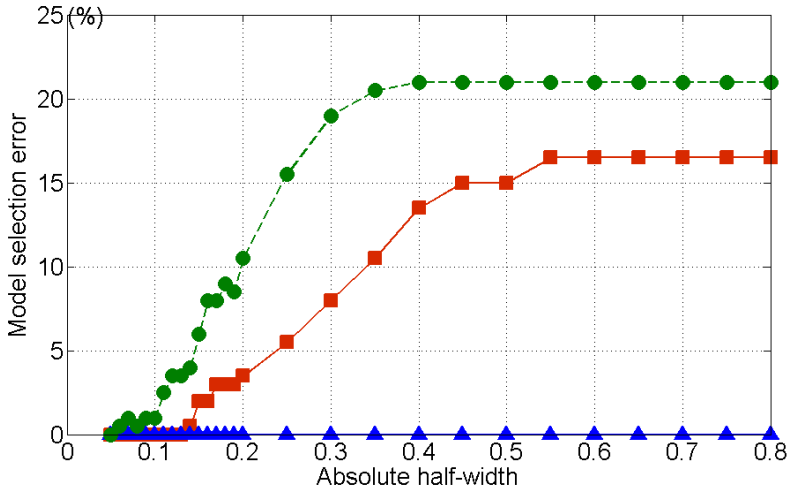


Figure 4.12: Error rate for standard RJMCMC (red, square) and SSD-RJMCMC (blue, triangle) methods for the synthetic data. The frequency of oberserving multiple possible solutions for global model determination in SSD-RJMCMC (Case 2 in Stage 2) is plotted as a green dashed line, marked with circle.

the standard serial RJMCMC with the longest triple-state RJMCMC chains in the SSD-RJMCMC. When tightening the convergence bound, the concurrent sequences in SSD-RJMCMC grow aggressively, but are always shorter than the RJMCMC chains. When AHW_k equals 0.2, the longest SSD-RJMCMC chains are more than one third shorter than the RJMCMC chain, which greatly reduces the computation complexity. Under this condition, as plotted in Figure 4.12, about 4% of the RJMCMC chains have the model selection error, giving incorrect inference of the surface number. By contrast, SSD-RJMCMC has zero error rate for this synthetic data set, although 10% of the trials observe multiple possible solutions for global model determination, the Case 2 of Stage 2. This implies that the existence of the local model selection errors, and demonstrates the effectiveness of the error correction procedure.

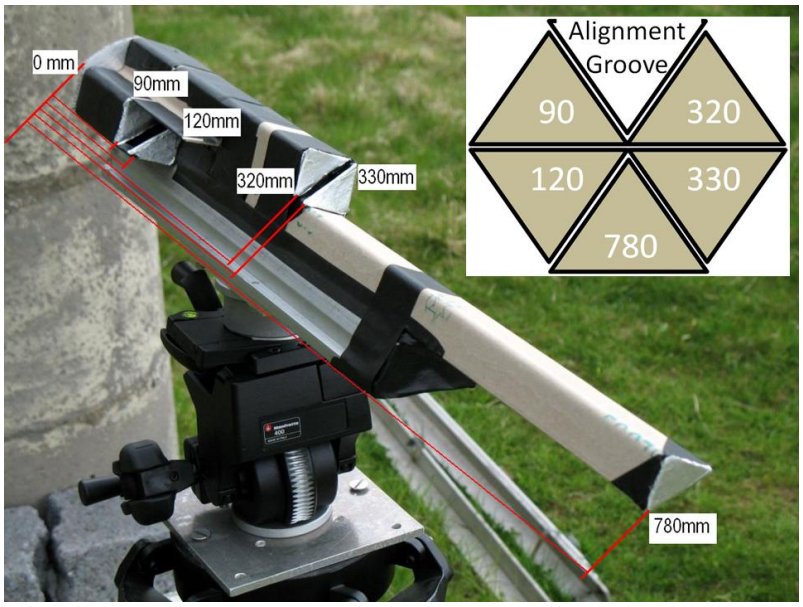


Figure 4.13: The target with multiple surfaces scanned at a distance of 325m in daylight and a sketch of the groove alignment. The total length of the scene was about 780mm from the nearest triangular surface to the back board. The profile lengths are relative to the back board (the target depth) measured in mm.

4.4 Experimental Evaluation on Real Data

We now consider the primary application to measurement of depth profiles from full waveform LaDAR, identifying improvements in *efficiency*, *reliability* and *accuracy* in comparison with serialC and parallel MCMC methods. The measured target shown in Figure 4.13 is a cardboard V groove with known alignment. The front of the triangles was covered with tin foil to maximize photon flux. The data was acquired in bright daylight at a range of approximately 325 meters. The pulse repetition frequency was 3MHz, resulting in an averaged optical power of 50μW. The bin separation in each histogram of photon counts was 4ps. Because of the area of laser footprint, many pixels observed multiple reflections corresponding to the distributed surfaces. An example of a trial measurement is shown in Figure 4.14.

For each method, we performed 100 independent runs and reported the average perfor-

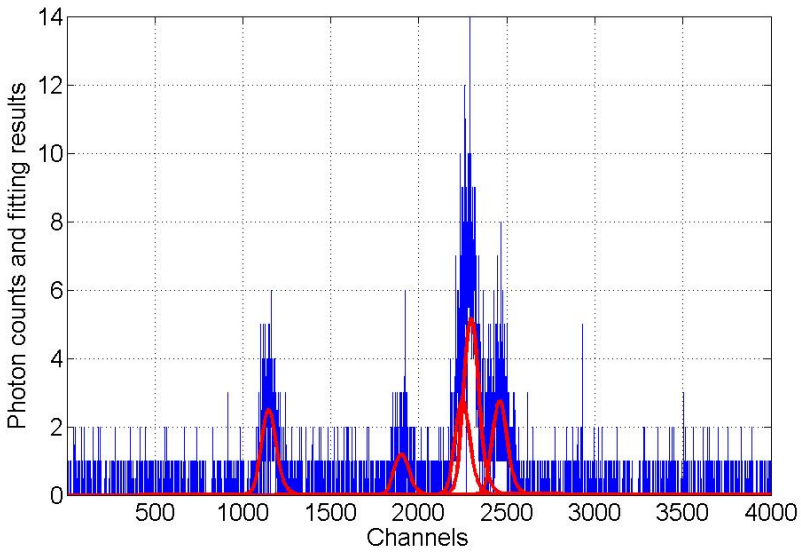


Figure 4.14: Time-of-flight LaDAR histogram (blue) on a single pixel from the remote distributed target shown in Figure 4.13 and final fitting results (red).

mance. Since the performance is closely related to the convergence bound in k, we assessed performance using different AHW_k values, specified as loose or tight depending on whether $\text{AHW}_k \geq 0.2$ or $\text{AHW}_k < 0.2$ respectively. The AHW threshold for t_0 is still fixed at 5.

4.4.1 Comparison of Efficiency

We consider firstly the efficiency of parallel MCMC chains. Here, we stopped each chain after 10,000 iterations and diagnosed the convergence on t_0 every 100 samples after the first 300 iterations. As shown in Table 4.2, with k fixed at 5, only 3.0% of the trials pass the convergence test (both stationarity and AHW tests in HWD) on all t_0 components. Regarding the other 97.0%, there is convergence on average to 58.6% of the t_0 components, about 3 of 5 positions, while the remaining two components have not escaped from the burn-in periods. When $k = 3$, 68.0% of the chains pass the convergence

	Percentage of chains that pass convergence test	Average burn-in length for converged chains	Average chain length for converged chains
k=3	68.0%	303	961
k=5	3.0%	210	717
k=7	0	NA	NA

Table 4.2: Convergence assessment on t_0 for parallel MCMC chains method using real data.

tests, while for $k = 7$, no MCMC chain achieves convergence because some t_{0_j} values are a result of false returns and hence never converge. Referring to Section 3.6, in the parallel MCMC chains method, within-model parameter estimates are the preconditions for Bayesian model selection. Without converged t_0 estimates, it is not valid to analyze LaDAR signals containing multiple returns. Therefore, the parallel MCMC chains method is not considered further.

For convergence assessment of k in reversible jump chains, we set both N_0 and ΔN to be 500. Figure 4.15 compares the average convergence length of the longest chain in the SSD-RJMCMC and serial RJMCMC methods as a function of AHW$_k$. For example, with loose convergence, the length of the chains in SSD-RJMCMC is reduced by up to 36.8% compared with serial RJMCMC. With tight convergence, the convergence length in both samplers increases dramatically. Since AHW is proportional to $\sqrt{\widehat{S}(0)/n_p}$, and a tighter convergence bound improves between-model exploration and results in a more stable $\widehat{S}(0)$ value, the sample size n_p must grow significantly to achieve the small AHW$_k$ bounds. Hence, although our method still achieves a similar 35.5% sample reduction, the efficiency improvement regarding the absolute chain length decrease is most marked under tight convergence.

Figure 4.16 contrasts the required chain lengths to achieve the same reliability levels for model selection. For example, serial RJMCMC needs 6335 samples for a zero frequency of overall model selection error (error rate) when AHW$_k$ equals 0.14; while the longest

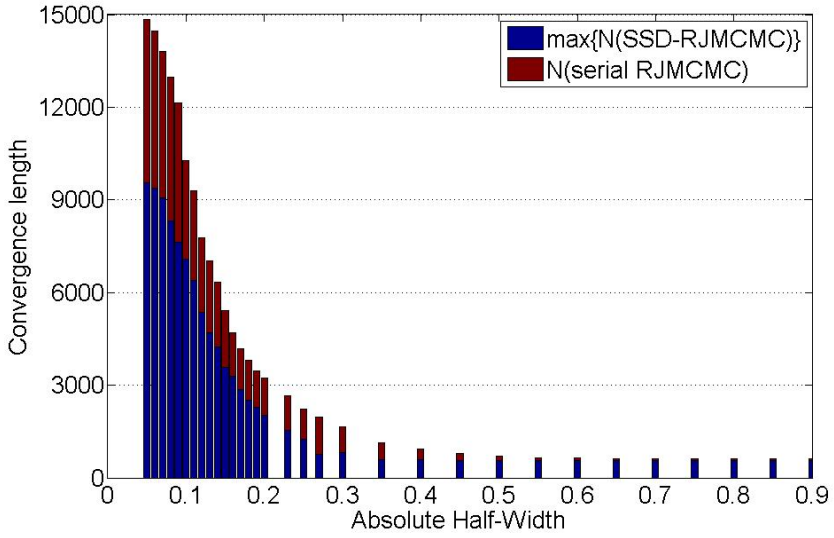


Figure 4.15: Average convergence length of the longest chain in SSD-RJMCMC (blue bar) and standard RJMCMC (red bar) methods for the real data using the same AHW_k values.

parallel triple-state RJMCMC chain needs 2040 samples, 67.8% chain length reduction, when AHW_k equals 0.2. For these two cases, Table 4.3 shows the corresponding chain lengths and timing results. In SSD-RJMCMC, the processing time for the longest chain is about 3 times shorter when compared to standard RJMCMC.

4.4.2 Comparison of Reliability

There are six distinct surfaces, but the histogram resolution is insufficient to resolve the 320mm and 330mm surfaces, so we consider five peaks as the "correct" answer. Figure 4.17 presents the error rate of k (denoted as ϵ_k), our measurement of reliability, in the serial RJMCMC and SSD-RJMCMC methods.

First, compared to serial RJMCMC, ϵ_k in the SSD-RJMCMC method starts to vary with convergence bound only when AHW_k is less than 0.5. As asymptotic variance of the sample mean in the triple-state RJMCMC chains is largely confined by the reduced model

	Time/iter(ms)	Averaged convergence length for zero ϵ_k	Averaged proceeding time (sec)	%
RJMCMC	4.542	6335	28.776	100
chain_0	2.250	500	1.125	3.91
chain_1	2.710	515	1.396	4.85
chain_2	3.137	540	1.694	5.89
chain_3	3.816	755	2.881	10.01
chain_4	4.179	1180	4.932	17.14
chain_5	4.489	2040	9.157	31.82
chain_6	4.951	1160	5.743	17.92
chain_7	5.179	845	4.376	15.21

Table 4.3: Convergence lengths and execution times for the RJMCMC and SSD-RJMCMC chains.

space, the use of a loose convergence bound does not affect model selection reliability.

Second, using an effective loose convergence bound, $0.2 \leq \text{AHW}_k < 0.5$, besides the chain length reduction (see Figure 4.15), our approach achieves a significant reliability improvement. In particular, the maximum ϵ_k of serial RJMCMC up to 25.0% is reduced to 5.0% in the proposed framework. Moreover, in both samplers, ϵ_k declines rapidly when tightening the AHW_k bound from 0.5 to 0.2, but this bound has to be reduced more substantially to 0.14 in the RJMCMC method to approach zero ϵ_k, compared to 0.2 in SSD-RJMCMC.

Third, Figure 4.17 shows the frequency of detecting multiple possible solutions in Stage 2 of the proposed framework. This is higher than ϵ_k in the serial RJMCMC and SSD-RJMCMC methods. In the SSD-RJMCMC framework, some of the concurrent chains can have a high frequency of local model selection error, such as chain_3, chain_4 and chain_5. However, as shown in the lowest (blue, triangles) plot of Figure 4.17, after the second stage, the final error rate in global selection is much lower in the SSD-RJMCMC case. By contrast, without error detection and correction, serial RJMCMC is unable to recognize the local convergence and hence has a higher ϵ_k.

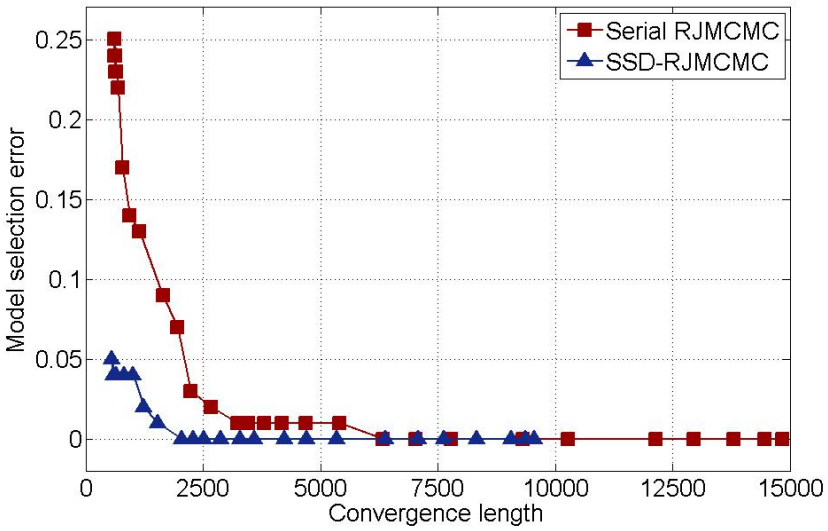


Figure 4.16: Error rate changes with convergence length in serial RJMCMC (red, marked with triangle) and SSD-RJMCMC (blue, marked with square) methods.

4.4.3 Comparison of Precision and Accuracy

For each serial RJMCMC chain in the repeated trials, we considered the longest segment for the selected model and assessed the convergence of t_0. The longest segment contained 277 samples on average, 94.1% of which passed the burn-in test with an average burn-in length of 16. This demonstrates that between-model jumps can help to locate the peaks within parameter subspaces, giving very short burn-in periods in the RJMCMC segment, particular in comparison with parallel MCMC chains method. However, due to the restricted segment length, only 16.0% can achieve the AHW_{t_0} bound. This indicates that even though k has converged, t_0 requires more consistent within-model updates to meet a certain level of estimation accuracy. SSD-RJMCMC performs similarly, but since it chooses the longest segment among all the chains containing the determined model, the averaged longest segment length is larger, and thereby is much more likely to pass the convergence test. Using complementary MCMC generation with about 343 samples on average in the third stage, all the t_0 components converge.

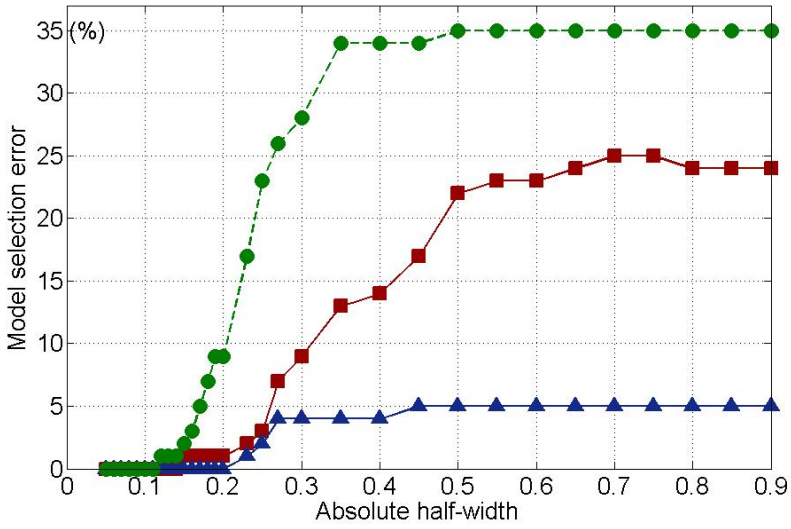


Figure 4.17: Error rate for standard RJMCMC (red, square) and SSD-RJMCMC (blue, triangle) methods for the real data. The frequency of oberserving multiple possible solutions for global model determination in SSD-RJMCMC (Case 2 in Stage 2) is plotted as a green dashed line, marked with circle.

	Separation 1	Separation 2	Separation 3	Separation 4
Ground Truth	455	205	30	90
RJMCMC	454.9	205.9	28.5	93.9
SSD-RJMCMC	455.8	204.0	28.8	95.2

Table 4.4: Comparison of actual target separations (mm.) and the measured mean estimates from 100 trials; each channel in the histogram corresponds to 0.6 millimeters.

Figure 4.18 and Table 4.4 show the comparative precision (in the form of a distribution of trial results) and accuracy (as mean estimates) of the adaptive concurrent method in comparison with the serial RJMCMC technique. This comparison assumes reliable convergence, which has been shown to be better in the adaptive case. Given a reliable measurement, the comparison with ground truth shows accuracies of 1-5mm at 325m range in each case, and comparable standard deviations that indicate the repeatability of the measurement. Although the two methods show very similar accuracy and precision, the advantage of the SSD-RJMCMC technique is the ability to more rapidly refine the within model estimates at Stage 3, on conclusion of a restricted value or set of values for k.

(a) t_{0_1}: (1148.9, 2.7) (b) t_{0_2}: (1907.2, 4.4) (c) t_{0_3}: (2250.3, 16.9) (d) t_{0_4}: (2297.8, 13.2) (e) t_{0_5}: (2454.3, 3.4)

(f) t_{0_1}: (1148.9, 2.8) (g) t_{0_2}: (1908.6, 4.4) (h) t_{0_3}: (2248.6, 14.1) (i) t_{0_4}: (2295.4, 9.0) (j) t_{0_5}: (2454.1, 3.0)

Figure 4.18: Distribution of trial results in serial RJMCMC method (Figure 4.18(a) to 4.18(e)) and SSD-RJMCMC method with complementary MCMC samples (Figure 4.18(f) to 4.18(j)). The two statistics for each plot are the mean and standard deviation of the t_{0_j} estimate.

4.5 Conclusions

In this chapter, we have presented and evaluated an adaptive concurrent framework, the SSD-RJMCMC algorithm, which draws on the complementary advantages of the serial RJMCMC and the parallel MCMC approaches for model selection and parameter estimation. First, we decompose the entire Markov state space and generate concurrent, independent RJMCMC chains, each with a much reduced variation in model dimension. Second, we incorporate convergence diagnostics to achieve dynamic chain length control. Third, we employ a *global* model determination scheme by comparing the separate *local* results from the concurrent chains. This allows us to detect possible errors in local model selection and provide correction. Fourth, based on convergence assessment of the parameters, we adaptively generate a complimentary MCMC chain for the chosen model(s) to obtain accurate parameter estimates.

Experimental results from both synthetic and real data acquired by our own LaDAR system have shown improvements in sampling efficiency and model selection reliability, together with comparable accuracy and precision of parameter estimation, when compared with other two methods. First, as shown in Section 4.4.1, compared with standard RJMCMC, the Markov chain length of the triple-state RJMCMC chains is considerably reduced, since the between-model mixing difficulty of exploring the entire space is now shared among a set of independent sequences with diminished variation of model dimension. Second, as present and discussed in Section 4.3.2, the re-configured state space ensures that all the candidate models are properly explored, addressing the problems of escaping from local optima and concluding local convergence in conventional RJMCMC samplers. Further, based on the layout of $\{\widehat{K}_{\text{chain}_i}\}$, the designed global model selection scheme helps to detect any local model selection error and provides a chance for further model comparison. A local error is corrected in the succeeding stages. As shown in Section 4.4.2, both factors lead to an improved reliability in comparison with standard RJMCMC. Third, the accuracy of parameter estimation can be enhanced by a fuller repre-

sentation of the model-specific parameter posterior. This complimentary MCMC chain is not compulsory, but is performed optionally depending on the convergence performance in the RJMCMC segments and desired accuracy level for parameter estimates.

Although the concurrent sampling in SSD-RJMCMC can naturally support parallel processing, filling the vacant area of RJMCMC parallelization, it requires future work for parallel implementation on a distributed cluster. We need to consider some practical issues affecting the parallel performance, notably load balancing between triple-state chains since the chain lengths are different in each model group, and designing efficient parallel programs to avoid frequent communications between different processors.

Chapter 5

Parallel Bayesian Inference of Surface Range and Reflectance from LaDAR Profiles

5.1 Introduction

Although RJMCMC is particularly suited to estimate the number, range and reflectance of remote surfaces sensed by a LaDAR, the intensive and time-consuming computation can inhibit the scope of application. Clearly, parallel computing systems, themselves built from multi-core processors, offer the opportunity for major reductions in computation time at a reasonable cost. However, as discussed previously, because a Markov chain is essentially a serial process, where each estimate depending on the previous one, it is not straightforward to parallelize MCMC and RJMCMC algorithms. The first challenge is to use the conditional independence structure of the underlying models to divide the sampling procedure into a number of independent tasks. The second challenge is to use diagnostics of the mixing performance and the convergence length of the Markov chain to determine the sampling efficiency and the final accuracy of parameter estimations, that

is to divide and schedule the exploration of the parameter space to achieve an equivalent result in the standard serial processsing.

To overcome the statistical challenges, we have proposed a variant of the RJMCMC algorithm, State Space Decomposition RJMCMC (SSD-RJMCMC), which generates multiple independent RJMCMC chains with restricted variation in model dimension. This intrinsically supports parallel processing, and has the benefit of reducing the Markov convergence length of the resulting concurrent sequences. However, considering the MPI implementation in the computing domain, it has redundant message passing. Moreover, the basic algorithm also has difficulties to achieve load-balancing, and is not easily adapted to different numbers of processors.

This chapter investigates a new parallel RJMCMC algorithm that reduces computation time but maintains the estimation accuracy for LaDAR signal analysis. Our starting point is SSD-RJMCMC. To remove the redundant communications overhead, we firstly arranged the original framework. The updated version is termed as SSD-RJMCMC^{U}. To address the load-balancing and scalability problems, we have combined parallel data decomposition with the job scheduling of the SSD-RJMCMC^{U} framework, which we term as the data parallel SSD-RJMCMC^{U} (DP SSD-RJMCMC^{U}) method. This formalises a task queue and dynamically allocates the smaller-sized tasks to idle processors. It inherits the statistical benefits of SSD-RJMCMC, but reduces the load imbalance and can scale to larger numbers of processors.

5.2 Modified SSD-RJMCMC

5.2.1 Why Modify SSD-RJMCMC?

The concurrent sampling in SSD-RJMCMC naturally enables parallel processing. As a variant of serial RJMCMC, we demonstrated for LaDAR that SSD-RJMCMC brings statistical benefits in sampling efficiency, improved reliability of peak detection and maintaining the accuracy of surface reconstruction. In particular, the considerably reduced length of the concurrent RJMCMC chains offers the potential for speedup in parallel processing. Despite of these statistical advantages, the original framework can be modified to further enhance the processing efficiency when implemented on a parallel platform. Using SSD-RJMCMC as a prototype, we introduce SSD-RJMCMC^{U} with the following arrangement.

- In SSD-RJMCMC, we adaptively generate a complimentary MCMC chain for the chosen model to provide a more thorough exploration within the model-specific parameter subspace. This succeeding stage is not compulsory, but is performed optionally depending on the convergence performance and the desired accuracy level for the parameter estimates. For the sake of processing efficiency, we skip it to avoid the redundant computation and implementation complexity, when we do not require a higher estimation accuracy and precision than in RJMCMC.

- Suppose Stages 3 and 4 in SSD-RJMCMC are implemented in parallel, the master and workers must cooperate to extract the parameter estimate for the uniquely determined model or accomplish the error correction for the ambiguous models. Figure 5.1 illustrates the latter case, whilst the former one only conducts step(2) in the figure. This process is inefficient. First of all, it introduces additional inter-processor communications, four rounds of point-to-point message passing for the determined model or one of the ambiguous models. Moreover, all the workers must save the

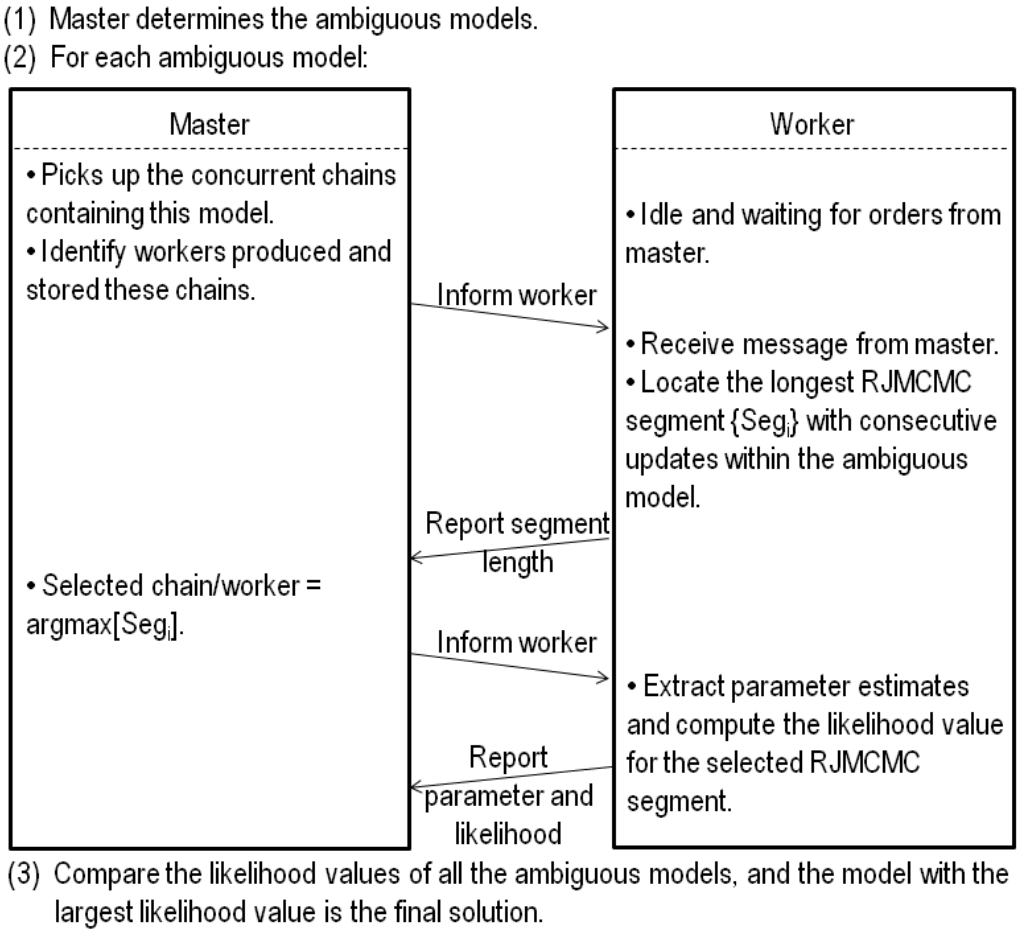


Figure 5.1: Error correction process in original SSD-RJMCMC. Seg_i is the i^{th} data segment.

generated samples until the method completion, which raises the requirement for large memory storage. Finally, the message passing and instruction execution are progressed sequentially, which results in waiting time in idle workers and aggravates the load imbalance. Therefore, in SSD-RJMCMC$^{\text{U}}$, each worker sends back the parameter estimates and the likelihood value to the master, and then frees the allocated memory space immediately.

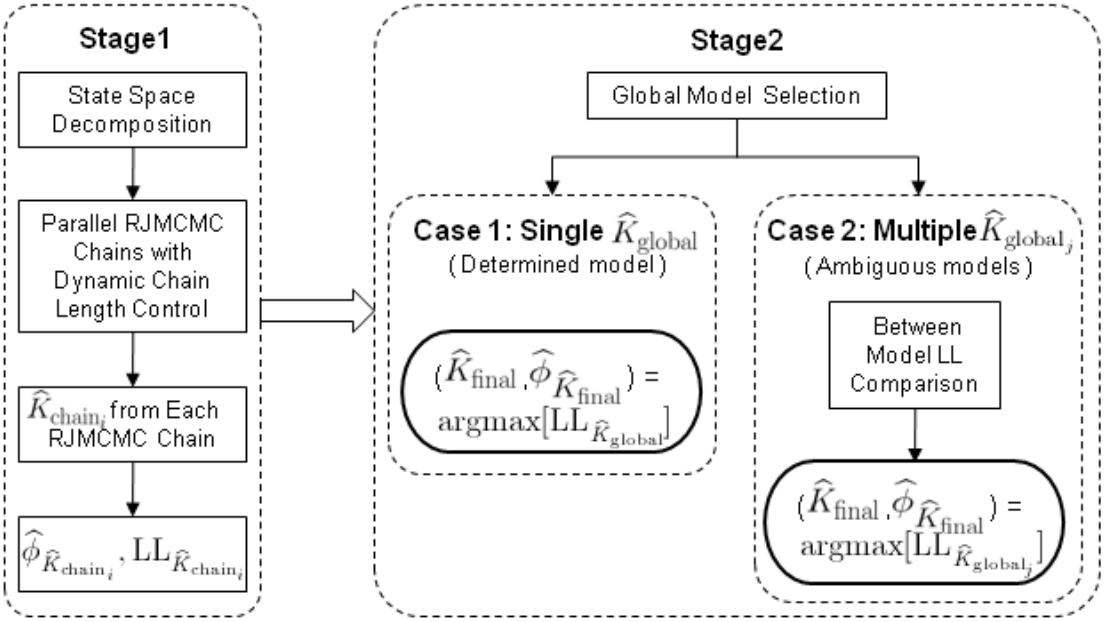


Figure 5.2: System diagram of SSD-RJMCMC$^{\text{U}}$ methodology.

5.2.2 SSD-RJMCMC$^{\text{U}}$ Framework

The SSD-RJMCMC$^{\text{U}}$ approach is divided into two separate stages as shown in Figure 5.2. Stage 1 implements concurrent RJMCMC sampling, generating a set of independent RJMCMC chains, each exploring three neighbouring models; conducts the local Bayesian model selection; and sends the local results to the master processor. Stage 2 makes a global model selection, and resolves any ambiguities of model selection, providing an error detection and correction scheme for the inference on k.

Stage 1: State space decomposition and local model selection

Similar to Stage 1 in SSD-RJMCMC (see Figure 4.2), for the concurrent sampling in an RJMCMC framework, the complete state space of n candidate models, $\{k_1, k_2, ..., k_n\}$ is divided into $(n-2)$ groups, each containing 3 adjacent models. Each group is assigned with an independent RJMCMC chain to explore the restricted models. Still applying the Heidelberger and Welch convergence diagnostic (HWD), the chain length is dynamically

controlled. For the converged chain, we conduct a *local* Bayesian model selection and obtain $\{\widehat{K}_{\text{chain}_i} : i = 1, 2, ..., n-2\}$.

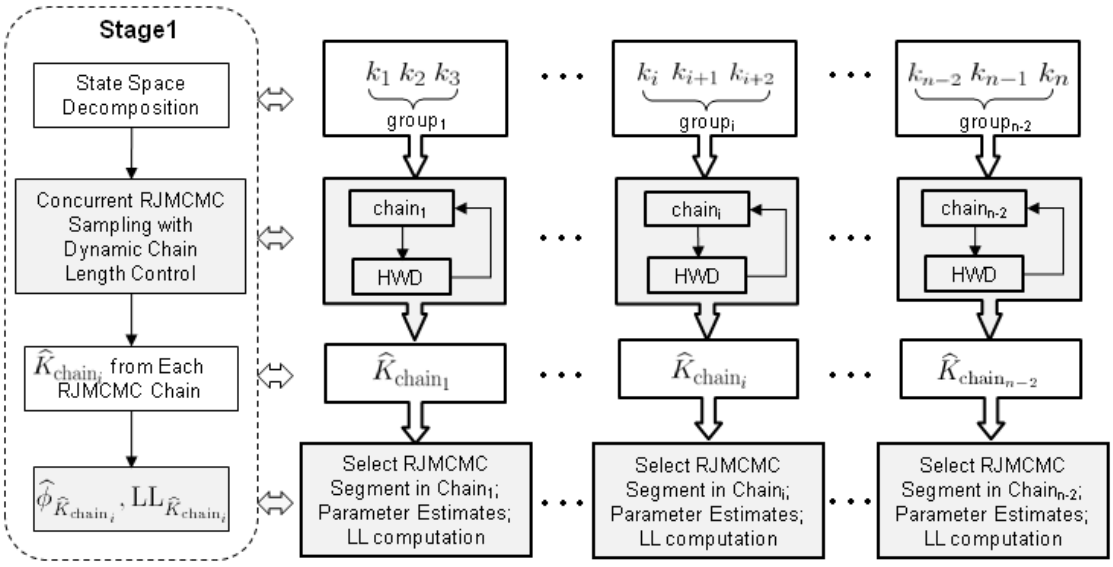


Figure 5.3: Stage1 in SSD-RJMCMC$^{\text{U}}$ algorithm.

The additional procedure in SSD-RJMCMC$^{\text{U}}$ is that once we have determined the number of peaks in each chain, we extract the parameter estimates $\widehat{\phi}_{\widehat{K}_{\text{chain}_i}}$ using the longest RJMCMC consecutive segment (sample trajectory) with k equal to $\widehat{K}_{\text{chain}_i}$. Then, we compute the log-likelihood, $\text{LL}(\mathbf{y}|\widehat{\phi}_{\widehat{K}_{\text{chain}_i}}, \widehat{K}_{\text{chain}_i})$.

Stage 2: Global model determination

Stage 2 in SSD-RJMCMC$^{\text{U}}$ combines Stages 2 to 4 in the original SSD-RJMCMC framework. First, with reference to Figure 4.3, we make a *global* model determination based on $\{\widehat{K}_{\text{chain}_i}, i = 1, 2, ..., n-2\}$ from the concurrent triple-state RJMCMC chains in Stage 1. Second, we define an error detection scheme by combining and analyzing all the *local* results. Generally, there are two cases:

- Case 1: *Single solution for global model determination.*

  Recalling Figure 4.3(a), if all triple-state RJMCMC chains pick up the correct

model, we have the unique model selection result as $\widehat{K}_{\text{final}} = \widehat{K}_{\text{global}} = k_i$. Accordingly, the parameter estimates are chosen from $\widehat{\phi}_{\widehat{K}_{\text{chain}_i}}$ in Stage 1 as the one that holds the largest likelihood value for $\widehat{K}_{\text{final}}$, i.e. $\widehat{\phi}_{\widehat{K}_{\text{final}}} = \text{argmax}[\text{LL}_{\text{chain}_i}|_{\widehat{K}_{\text{chain}_i}=\widehat{K}_{\text{final}}}]$.

- Case 2: *Multiple possible solutions for global model determination.*
  According to Figure 4.3(b), if one of the concurrent chains has a local model selection error, we observe ambiguous models, and they are the multiple possible solutions for global model determination. For each of them, we then find the within-model parameter estimates in the same way as in Case 1, giving $(\widehat{\phi}_{\widehat{K}_{\text{global}_j}}, \text{LL}_{\widehat{K}_{\text{global}_j}})$. Finally, we make the between-model comparison, and the model with the largest likelihood is the final solution, i.e. $(\widehat{K}_{\text{final}}, \widehat{\phi}_{\widehat{K}_{\text{final}}}) = \text{argmax}[\widehat{LL}_{\widehat{K}_{\text{global}_j}}]$.

5.2.3 The Challenges for Parallel Implementation

Although the structure of the SSD-RJMCMC$^{\text{U}}$ methodology can naturally support parallel processing, it is still challenging to implement this on a distributed multiple instruction multiple data (MIMD) computing cluster. The decomposed state space of Stage 1 produces a fixed number of model groups. This suggests a natural and efficient solution when we have an equal number of processors (P) and concurrent sequences (N). But, it is not straightforward as the chains may have different convergence lengths, i.e. the number of iterations per chain is not fixed. In addition, chains have different numbers of "fixed" parameters in each sweep; although updated as a vector, they do not have equivalent time complexity since more parameters bring in more computation in likelihood evaluation. For these two reasons, load imbalance is unavoidable. If we have $P < N$, the solution is to queue the independent chains, but this may be sub-optimum as we do not know the length of each chain in advance, as it is signal dependent. Conversely, if $P > N$, the extra processors are redundant and do not assist in sample generation, unless we introduce parallel processing of a single three-state chain. To summarise, load-balancing is a crucial

issue that is particularly apparent in SSD-RJMCMC^{U} when there are large differences of convergence length and computational complexity among the resulting Markov chains.

5.3 Data Parallel SSD-RJMCMC^{U}

In SSD-RJMCMC^{U}, Stage 1 contains nearly all of the computation, whilst Stage 2 is simple and fast ($9\mu s$ for Case 1, $12\mu s$ for Case 2). Therefore, we only implement Stage 1 in parallel. The implementation is on a distributed Beowulf cluster, with considerations of the inter-processor communication costs and the memory storage requirement on each individual machine. The rest of the section will describe the data parallel SSD-RJMCMC^{U} methodology (DP SSD-RJMCMC^{U}), and demonstrate how the data decomposition addresses the challenges stated in Section 5.2.3.

5.3.1 Data Parallelism

In SSD-RJMCMC^{U}, the division of the RJMCMC state space into model groups provides *model-level* parallelization. Here, we introduce the signal data decomposition, formalizing the *data-level* parallelization, by dividing the data into segments and performing independent Bayesian model selection on each segment. Our implementation combines these two levels and the computation tasks are mapped into a two-dimensional grid topology as illustrated in Figure 5.4.

The data set is evenly separated into s segments, and the state space containing n candidate models is divided into c groups, where $c = n - 2$ for the triple model group. For each data set, we apply the SSD-RJMCMC^{U} methodology, which generates c parallel RJMCMC sequences. Each sequence is defined as a task, denoted as $T_{i,j}$ where i indicates the model group (chain index) and j identifies the data segment (data index). This

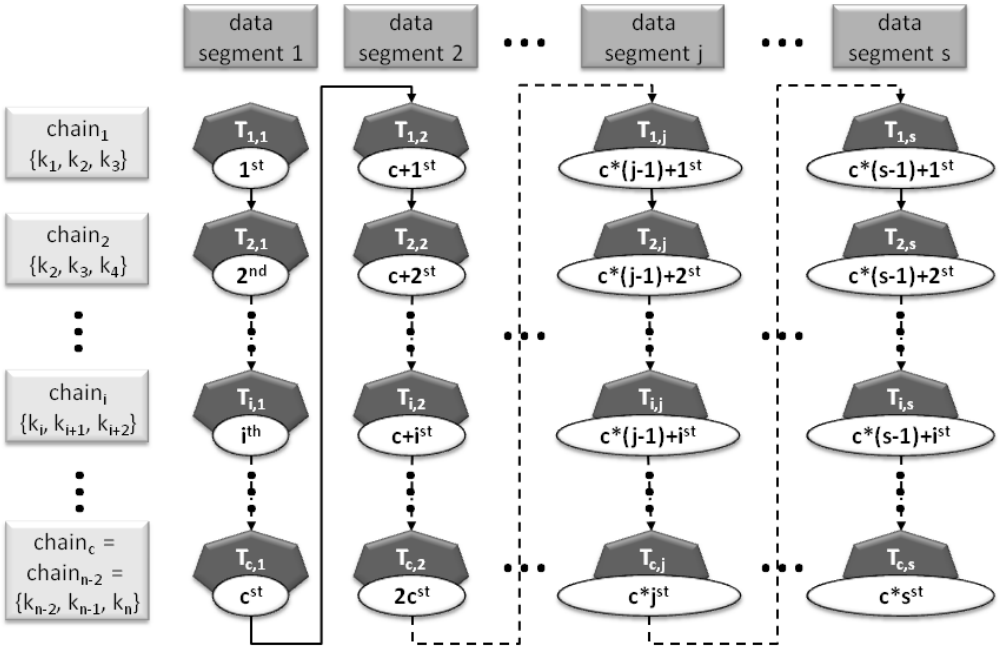


Figure 5.4: Two level parallelization.

creates a two-dimensional task pool with size $c \times s$, wherein all the tasks are mutually independent.

5.3.2 Dynamic Task Allocation

Data parallelism breaks down the tasks in SSD-RJMCMC^{U} into smaller chunks and generates a task pool. Although we can use the HWD diagnostic to monitor the convergence and conclude the task completion, the chain length is variable and not predictable. Therefore, a well defined scheduler and a dynamic task allocation scheme are crucial to achieve load-balancing. Figure 5.4 shows the matrix indexing for task assignment, where each task is indexed with a "task ID". This formalizes a task queue. When a processor finishes its current task, it queues to get a new one.

We use the master-slave model for parallel implementation, where the master has the

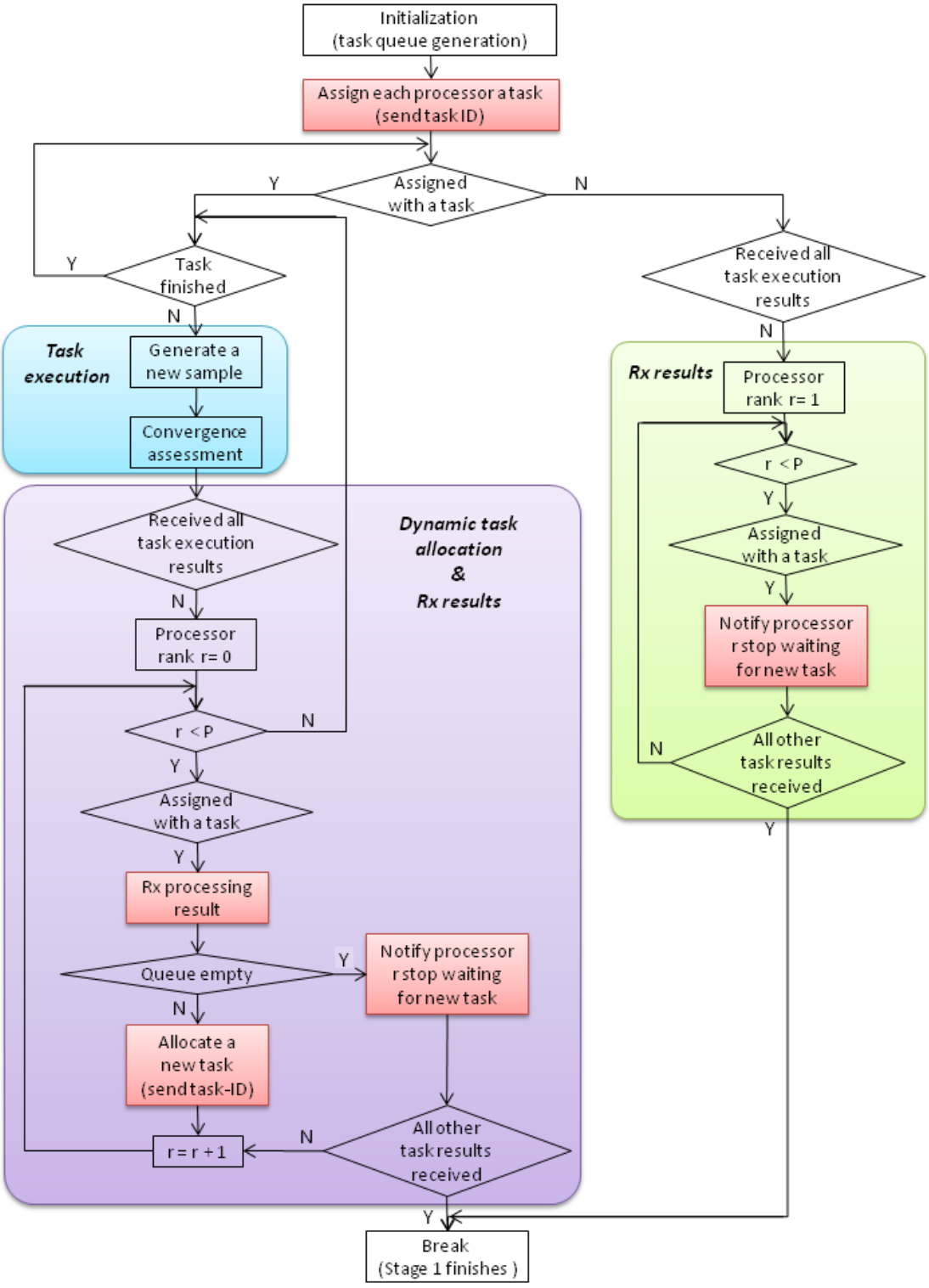


Figure 5.5: Workflow of the master processor in DP SSD-RJMCMC^{U}. (Rx = Receive.)

unidirectional control over all the slave processors (worker). The Master is responsible for dynamic task allocation, and meanwhile participates in task execution. It follows the workflow shown in Figure 5.5:

1. *Task execution and control process.*

   Initially, the master assigns each processor with a task, and activates communication monitoring for each worker. When executing the task, the master firstly generates a new sample, and then calls the control process after each updating sweep. The control process involves receiving the processing results from the workers and allocating the waiting tasks to the idle processors. It is a unified procedure for both the master and workers, while there is no inter-processor communication when dealing with the master itself. The master conducts the following procedure for all the processors:

```
1 if (processor assigned with a task and the task execution
    completes)
2     receive processing results from the processor
3   if (task queue is not empty)
4     assign a new task (send task ID)
5   else
6     send "queue empty" flag
7     if (all task results received)
8       chain generation (Stage 1) finished.
```

   The task completion for the master is defined by the chain convergence, while for a worker is defined by the status of the receive buffer, i.e. the buffer is filled with the processing results $(\widehat{K}_{T_{i,j}}, \widehat{\phi}_{T_{i,j}}, \mathrm{LL}_{T_{i,j}})$ sent from the worker. If the master is not executing a task, it indicates the task queue must have become empty. In this case, the master only receives results from workers and sends back a "task queue empty" flag to inform workers to stop waiting for new tasks.

2. *Data-level result analysis.*

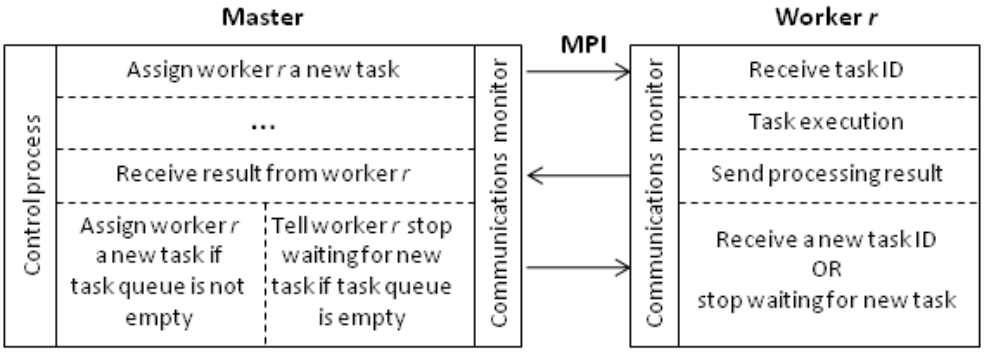


Figure 5.6: Commucation between the master and workers in DP SSD-RJMCMC^{U}.

The master follows Stage 2 in SSD-RJMCMC^{U} to retrieve peak information for each data segment.

3. *Final conclusion.*

   The master combines $(\widehat{K}_{\text{Seg.}_j}, \widehat{\phi}_{\text{Seg.}_j})$ for each data segment and draws a final conclusion for the complete histogram.

5.3.3 Communication

Inter-processor communications only occurs between the master and workers, and there is no direct message passing between any two workers. Figure 5.6 illustrates the communication between the master and a worker, r.

Initially, the master broadcasts the task IDs to all processors and initiates the non-blocking receive from each worker to obtain the processing results. It then keeps on monitoring the status of the receive buffer until it is filled. With accordance to the task queue status and the task receipt completion, the master either sends worker r a new task ID or "task queue empty" flag via non-blocking send.

The worker starts with an initially allocated task and initiates the non-blocking receive

to get instant orders from the master. After completing the task execution, it uses non-blocking communication to send back the processing results, and then receives a new task ID. The communication monitoring is repeated until receiving the "task queue empty" flag.

5.3.4 Method Discussion

Statistically, first, SSD-RJMCMC^{U} reduces the convergence lengths of the Markov chains in the model-level parallelization by restricting the dimensionality of the state space. DP SSD-RJMCMC^{U} holds the potential to further decrease the chain length in the data-level parallelization by simplifying the histogram subject to inference. Second, DP SSD-RJMCMC^{U} inherits the fundamental framework of SSD-RJMCMC^{U} and therefore is capable of detecting and correcting the model selection error caused by insufficient between-model mixing. Third, the data parallelism technique can give us some data segment containing incomplete system response. It is an open question that whether or not the algorithm can properly interpret the incomplete signals. This will be investigated in Section 5.4.

From the view of parallel computing, first, all the tasks are independent and there is no inter-processor communication until the task completion. This provides a coarse-grained components and avoids frequent message passing. Second, the load balancing problem can be solved by an automatic scheduler for dynamic task allocation based on the generated task queue with smaller and similar task size. Third, the algorithm has large scalability with processor number due to the large bucket size of the task queue.

The advantages in both the statistical and computing domains can help to improve the simulation efficiency. As illustrated in Figures 5.2 and 5.3, Stage 1 of the SSD-RJMCMC^{U} contains most of the computation. As discussed in Section 2.2.5, the computational complexity of the RJMCMC sampling is $O(NkL)$, where the chain length N is the most

crucial factor, relying on the data set. Taking timing results presented in Table 4.3 as an example, because of differences convergence length and the model dimensions (k), the standard RJMCMC takes 28.776 seconds while the serial implementation of the SSD-RJMCMC algorithm takes 31.304 seconds. In DP SSD-RJMCMC^{U} , the data-level parallelization affects the overall complexity by reducing the data size L, but the computation time of the serial implementation would be comparable with the RJMCMC and SSD-RJMCMC approaches. In addition, at the beginning of this section, we mentioned that the Stage 2 of the SSD-RJMCMC^{U} takes $9\mu s$ to $12\mu s$. Therefore, if we implement Stage 1 of the DP SSD-RJMCMC^{U} in parallel, we can expect nearly linear speedup. However, the inter-processor communication and load imbalance caused by the difference of the convergence length of the parallel sequences can degrade the parallel performance to a certain extent. Moreover, the performance is closely related with the configuration of the algorithm, i.e. the number of data segment s and the number of processors P.

5.4 Algorithm Performance Evaluation

In this section, we will evaluate the statistical effectiveness and parallel implementation efficiency of the DP SSD-RJMCMC^{U} methodology. For the statistical property, we will assess the convergence length and the processing time of the parallel chains in DP SSD-RJMCMC^{U}, and compare with the standard RJMCMC and SSD-RJMCMC^{U} methods. This will illustrate how data parallelism can affect the sampling efficiency and help to improve load-balancing. We also evaluate the parameter estimation accuracy and investigate whether the RJMCMC sampler can correctly resolve the surface return stepping across two neighbouring data segments. For the MPI implementation, we will discuss inter-process communications and measure the speedup and efficiency achievement.

The evaluation was carried out on a Beowulf cluster with 32 nodes, with configurations

Node	eight-core, Intel Xeon 2GHz CPUs
Operating system	Red Hat Linux 4.1.2-50
MPI version	MPICH 2 1.2.1p1

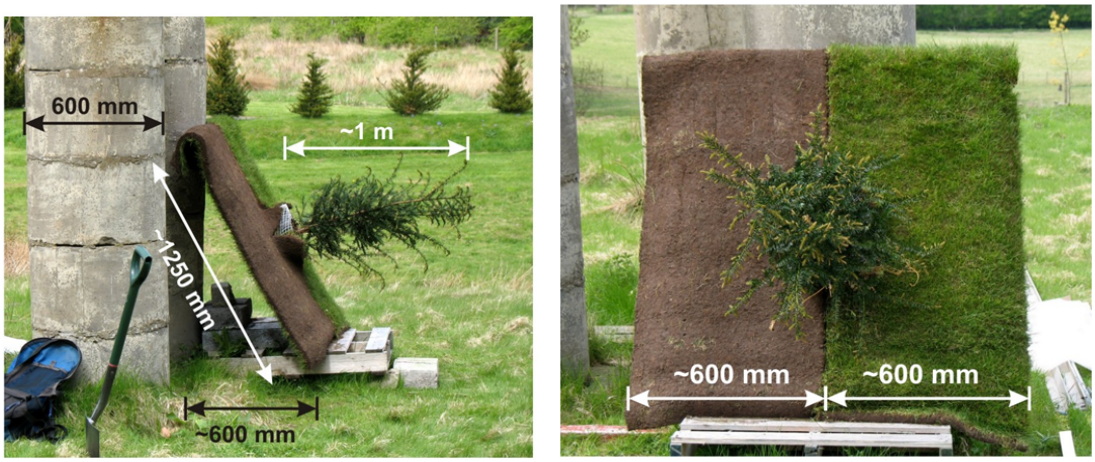
Table 5.1: Configuration of distributed Beowulf cluster.

shown in Table 5.1. In consideration of the inter-node communications overhead, we only used one CPU from each machine. Regarding the randomness in convergence length, computation environment and dynamic network traffic, we present the results from 100 repeat runs and compute the averaged statistic.

5.4.1 LaDAR Data and Sampler Setting

Our evaluation uses two data sets, one synthetic and the other real. The synthetic data (see Figure 4.4) is the one used to illustrate SSD-RJMCMC in Section 4.3. It provides the ground truth to determine whether or not the proposed method is able to infer the exact number of peaks. The real data is acquired from a horizontally mounted tree in front of a sloped surface covered with grass and clay. This data was collected as part of a separate programme to investigate the use of TCSPC LaDAR data to recover schedule and reflectance parameters from above tree canopies. The laser pulse repetition frequency was 3 MHz, and the illumination power at the target was approximately 50 μW on average. The scanned scene is shown in Figure 5.7. Scans were taken on a sunny day in broad daylight. Atmospheric conditions throughout all scans stayed relatively constant. The selected stand-off distance of about 325 metres results in a beam spot size at the target of approximately 14 mm. The bin separation in each histogram of photon counts was 16ps. Due to the area of laser footprint, many pixels observe multiple peaks from the distributed surfaces. The representative pixels are shown in Figure 5.8, while the measurement under our analysis is Figure 5.8(a).

To compare our approach with conventional RJMCMC and SSD-RJMCMC$^{\text{U}}$ algorithms,



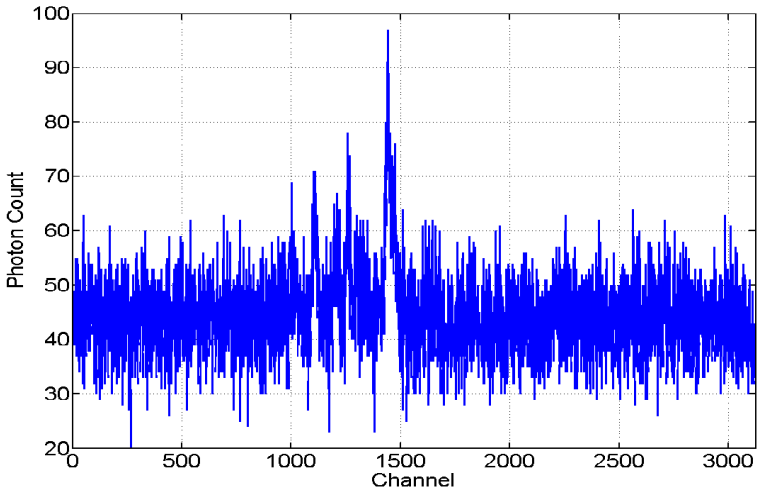
(a) (b)

Figure 5.7: The real target: a horizontally mounted tree in front of a sloped surface covered with grass and clay.

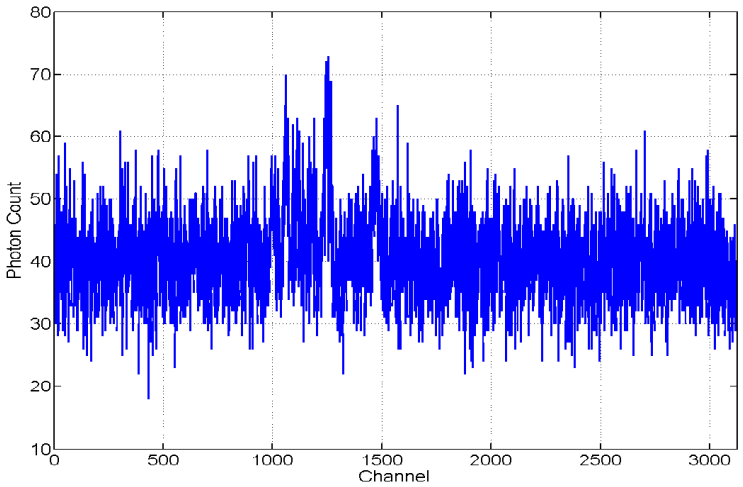
we ensure consistency of sampler design in all experiments. For the prior probability defined in (2.10), we set the previously unspecified constants as: $a = 1.5$, $b = \text{mean}(y)$, $c = 1.03$ and $d = 1000$. The value of a and b have been chosen to allow the detection of possible small returns embedded in the background noise, whereas c and d have been chosen to represent vague prior knowledge of the background.

The within-model parameter updates in moves (1) to (3) (defined in Section 2.2.3.4) follow Gaussian random walk proposals, whose means are the current sample values, with standard deviations 10 bins for t_0, and 0.3 counts for β and B. These are heuristic choices, based on recommended levels for proposal acceptance [101]. The between-model jumps in the RJMCMC sampling performed in the same way as defined in Section 2.2.4.

In testing, we constrained the possible number of peaks from 0 to 9 in RJMCMC for both data , giving eight triple-state chains (chain_0 to chain_7) in SSD-RJMCMC$^{\text{U}}$. For the DP SSD-RJMCMC$^{\text{U}}$ application on the synthetic data, we divide the histogram into four segments. In each segment, although there are less peaks, we conservatively set the same range of k, in order to compare the convergence length and the processing time of



(a)



(b)

Figure 5.8: Time-of-flight LaDAR histograms on a single pixel from the remote distributed target shown in Figure 5.7.

the reversible jump chains in SSD-RJMCMC$^{\mathrm{U}}$ exploring the same model group but with different data sizes. This gives 4×8 tasks in total, and the starting value of k in each task was random.

To assess the parallel performance under different settings, we divide the entire histogram of the real data into 2, 4, 6 and 8 segments. When the number of segments s is set

to 1, 2 and 4, we choose the range of k to be suitably large in DP SSD-RJMCMC$^{\mathrm{U}}$, generating 8, 16 and 32 tasks respectively. In this scenario, the results presented for MPI implementation are the worst-case performance, and any prior knowledge can further improve the simulation efficiency by reducing the k range in each data segment. For instance, considering the limit size of our cluster and the shorter chunks of histogram can reasonably contains less peaks, we set the range of k to be 0 to 6 for $s = 6$, and 0 to 5 for $s = 8$, which gives 30 and 32 tasks respectively. Table 5.2 summarizes the experimental settings.

Data	Number of data Seg.	Range of k per Seg.	Total number of tasks
Synthetic data	1	$[0, 9]$	8
	4	$[0, 9]$	32
Real data	1	$[0, 9]$	8
	2	$[0, 9]$	16
	4	$[0, 9]$	32
	6	$[0, 6]$	30
	8	$[0, 5]$	32

Table 5.2: Experimental settings for DP SSD-RJMCMC$^{\mathrm{U}}$.

For dynamic chain length control, we applied HWD and analyzed the AHW statistic defined in Section 4.2.5. The significance level α was set as 0.05, and the AHW threshold for k was $\mathrm{AHW}_k = 0.10$. For RJMCMC, SSD-RJMCMC and DP (SSD-RJMCMC)$^{\mathrm{U}}$ algorithms, we applied HWD every $\Delta N = 1000$ iterations after a minimum chain length of $N_0 = 1000$. As demonstrated in Chapter 4, the triple-state chains in SSD-RJMCMC have shorter burn-in periods and convergence lengths; therefore, we can actually decrease ΔN and N_0 to a certain extent. For the synthetic data only, we reduce both values in SSD-RJMCMC and DP SSD-RJMCMC$^{\mathrm{U}}$ to 200, which can still support a reliable convergence assessment based on our initial trials. This setting helps to highlight the differences of convergence length among the model groups and demonstrate the load imbalance. Otherwise, if N_0 is too large, many triple-state chains can converge even before it and then stop at the same length. However, for the real data, we kept on using the same convergence condition for all three methods, ΔN and N_0 both equal to 1000, to provide a convincible

and reasonable performance comparison.

5.4.2 Statistical Property: Sampling Efficiency and Task Granularity

DP SSD-RJMCMC^{U} is not a direct parallel implementation of the standard serial RJMCMC, but a variant sampling scheme with parallel tolerable framework. Its statistical properties inherently affect the efficiency of parallel processing. In particular, the mixing performance of the Markov chain largely influences the convergence length, which determines the task processing time and then load-balancing. This section will evaluate the sampling efficiency by comparing the convergence length (a measurement of task size) and processing time of RJMCMC, SSD-RJMCMC and DP SSD-RJMCMC^{U}.

5.4.2.1 Synthetic Data

To evaluate model-level parallelization, we compare the convergence length between RJMCMC and SSD-RJMCMC. First, Table 5.3 shows that the convergence length in standard RJMCMC is almost 3.7 times longer than the longest sequence in SSD-RJMCMC^{U}. This verifies that state space decomposition reduces the mixing difficultly and accelerates the convergence speed in the concurrent chains. Second, Table 5.4 allows an in-depth investigation. We noticed that chains exploring the true model (k_{46} and k_{57}) are the longest chains, which are considerably longer than the under-/over-fitting chains, and this takes approximately ten times longer processing time in the worst case. This reveals serious load imbalance and lack of synchronization in SSD-RJMCMC.

To assess the task size after applying the data-level parallelization, we made a comparison between SSD-RJMCMC and DP SSD-RJMCMC^{U}. First, for the same data segment,

	AHW $= 0.1$			AHW $= 0.15$		
	$N_{\max}$	T_{total}	ϵ	$N_{\max}$	T_{total}	ϵ
RJMCMC	11909	48.487	0%	6273	24.472	2%
SSD-RJMCMC	3219	31.779	0%	2400	23.881	0%
DP SSD-RJMCMC$^{\text{U}}$	2020	33.099	0%	1037	21.315	0%

Table 5.3: Averaged convergence length N and the error frequency ϵ in RJMCMC, SSD-RJMCMC and DP SSD-RJMCMC$^{\text{U}}$. N in SSD-RJMCMC and DP SSD-RJMCMC$^{\text{U}}$ is the maximum length among all the parallel sequences, ϵ is the overall performance after using the error detection and correction scheme in Stage 2.

chains exploring different model groups with different data parameters converge at different length. As highlighted in Table 5.4, the chains containing the true model still holds the largest length but downgrades to the model groups with lower dimensions, since the returned peaks are now distributed over data chunks. Second, because the number of peaks and the superposition of these peaks are different among the data segments, the longest chain length for each data segment is different. Third, the longest sequence in DP SSD-RJMCMC$^{\text{U}}$ is more than one third shorter than the longest one in SSD-RJMCMC. This demonstrates that the data parallelism relieves the difficulty of data analysis, and hence reduces the convergence lengths in DP SSD-RJMCMC$^{\text{U}}$. Thanks to the shorter convergence length, smaller value of k in the longest chains and less computation for data chunks, the task size becomes significantly smaller and similar, which dramatically diminishes the load imbalance. Fourth, for a particular model group, although both methods can explore the same state space, they perform different between-model mixing, which makes it difficult to compare the exact amount of computations from the number of samples. To provide a comprehensive impression of the total problem size, we add up the individual convergence length along each row in DP SSD-RJMCMC$^{\text{U}}$ and divide it by four, which gives the "equivalent" number of samples for the entire histogram in SSD-RJMCMC. We found that the total execution times of these two approaches are similar, but the differences of the virtual length and the virtual processing time along each row are obviously decreased compared with SSD-RJMCMC$^{\text{U}}$. This establishes the premise to handle the load balancing problem and improve the speedup.

	SSD-RJMCMC		DP SSD-RJMCMC$^{\mathrm{U}}$									
	Entire data ($k_{\text{true}} = 5$)		Seg. 1 ($k_{\text{true}} = 2$)		Seg. 2 ($k_{\text{true}} = 1$)		Seg. 3 ($k_{\text{true}} = 2$)		Seg. 4 ($k_{\text{true}} = 0$)			
	N	T	N_1	T_1	N_2	T_2	N_3	T_3	N_4	T_4	$\frac{\sum_{i=1}^{4}(N_i)}{4}$	$\sum_{i=1}^{4} T_i$
k_{02} (chain$_1$)	456	1.146	**420**	0.676	**649**	1.050	**600**	0.832	**915**	0.843	646	3.400
k_{13} (chain$_2$)	419	1.145	**1515**	2.022	**1125**	1.642	**2020**	2.980	506	0.788	1291	7.432
k_{24} (chain$_3$)	619	1.822	**1801**	2.627	620	0.908	**1211**	1.805	430	0.677	1015	6.0620
k_{35} (chain$_4$)	**874**	2.784	1020	1.436	506	0.753	658	0.986	506	0.795	672	3.970
k_{46} (chain$_5$)	**3219**	11.362	639	0.971	411	0.638	544	0.832	430	0.737	506	3.178
k_{57} (chain$_6$)	**1474**	5.455	468	0.726	344	0.543	439	0.696	506	0.834	439	2.799
k_{68} (chain$_7$)	1219	4.938	430	0.777	363	0.597	477	0.787	400	0.682	418	2.843
k_{79} (chain$_8$)	710	3.124	458	0.742	363	0.58	506	0.910	496	0.892	456	3.124
$\sum_{j=1}^{8} N_{\text{chain}_j}$	8990	31.776	6751		4379		6456		4189		5444	29.630
Total time		31.779										33.099

Table 5.4: Averaged convergence length N and processing time T (in second) of the parallel sequences in SSD-RJMCMC and DP SSD-RJMCMC$^{\mathrm{U}}$ algorithms for the synthetic data, with AHW $= 0.1$. The highlighted figures in bold font correspond to the convergence length of the chains containing the true model.

5.4.2.2 Real Data

Figure 5.9 displays the averaged convergence length and task execution time of concurrent chains in SSD-RJMCMC. Compared with standard RJMCMC (see Table 5.5), the maximum chain length in SSD-RJMCMC is reduced by 11 times, saving 94.16% of the task execution time. The serial processing of SSD-RJMCMC on a single machine takes 76.20% less of the processing time than serial RJMCMC. This demonstrates again that the triple-state chains with smaller variation in model dimensions can converge faster than standard RJMCMC. However, the difference of execution time among different tasks is still comparatively large. The consequence is the serious load imbalance when implemented on a parallel platform.

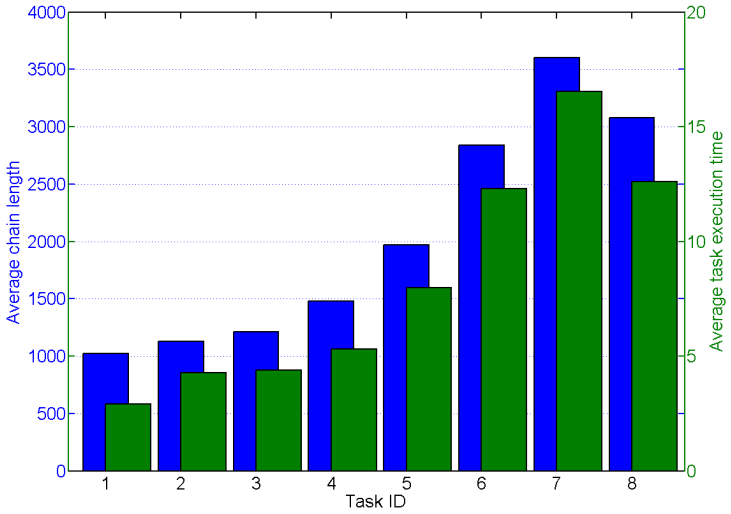
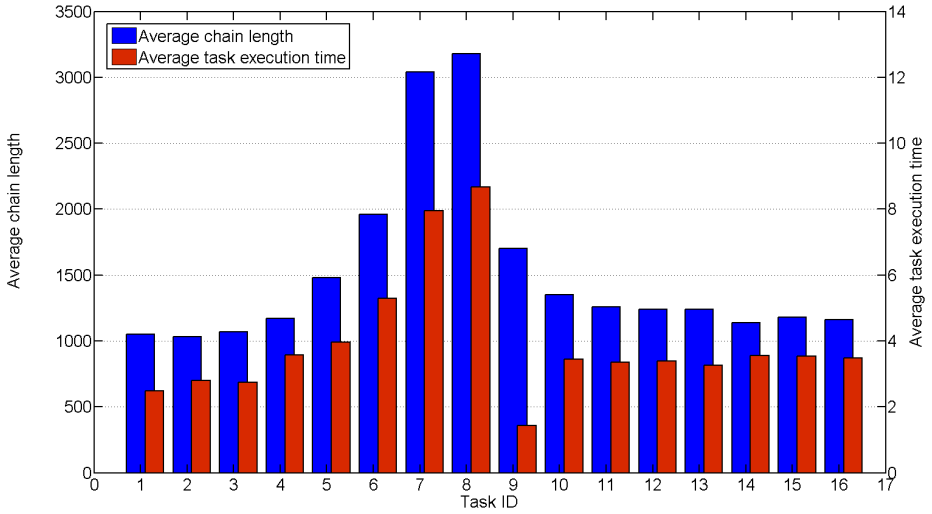
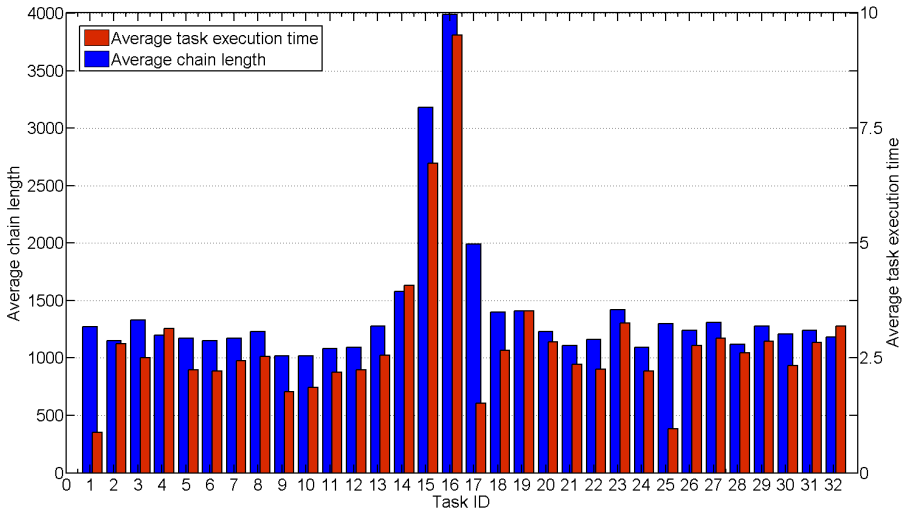


Figure 5.9: Average convergence length and task execution time (sec) in SSD-RJMCMC for the real data.

Results for DP SSD-RJMCMC$^{\mathrm{U}}$ with different numbers of segments are shown in Figure 5.10. We notice that when dividing the histogram into two separate chunks ($s = 2$), although the maximum convergence length is similar to SSD-RJMCMC, the longest processing time is almost decreased by halved. That is to say, in the generated task queue, the



(a)

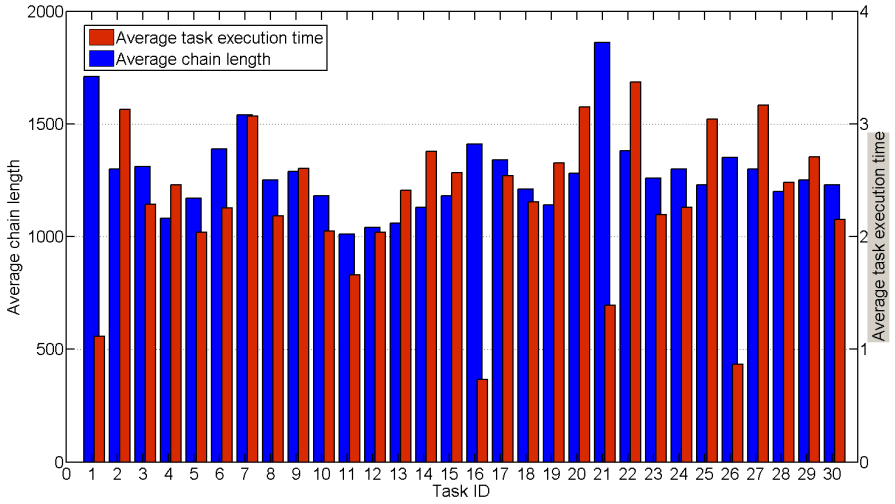


(b)

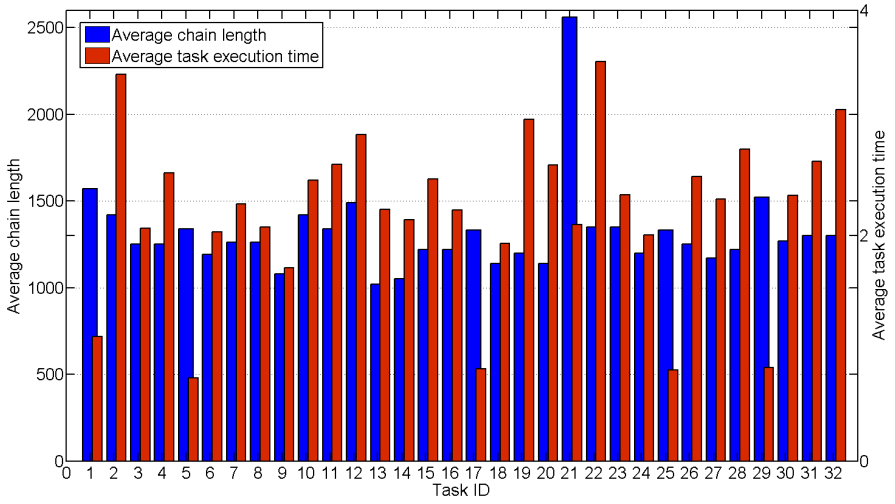
Figure 5.10: Average convergence length and task execution time (sec) in DP SSD-RJMCMC^{U} with 2 segments (a) and 4 segments (b).

bucket size is doubled, but the task size is halved. This brings benefits for load balancing.

When the LaDAR data is split into four segments ($s = 4$), the task queue is further extended. Comparing with the results for $s = 2$, the minimum execution time is nearly re-



(c)



(d)

Figure 5.10: (Continued)Average convergence length and task execution (sec) in DP SSD-RJMCMC^{U} with 6 segments (c) and 8 segments (d).

	$\min(N_i)$	$\min(T_i)$	$\max(N_i)$	$\max(T_i)$	N_{serial}	T_{serial}
RJMCMC					40303	283.346
SSD-RJMCMC	1020	2.898	3600	16.542 (5.8%)	16330	67.440
DP SSD-RJMCMC$^{\text{U}}$ ($s = 2$)	1030	1.429	3180	8.672 (3.1%)	24250	70.983
DP SSD-RJMCMC$^{\text{U}}$ ($s = 4$)	1020	0.883	3990	9.514(3.4%)	44600	96.050
DP SSD-RJMCMC$^{\text{U}}$ ($s = 6$)	1010	0.734	1860	3.372(1.2%)	38380	58.773
DP SSD-RJMCMC$^{\text{U}}$ ($s = 8$)	1020	0.737	2560	3.413(1.2%)	42010	78.348

Table 5.5: Comparison of the average convergence length and execution time (sec) in standard RJMCMC, and the average maximum chain length N_i and maximum task execution time T_i in SSD-RJMCMC and DP SSD-RJMCMC$^{\text{U}}$ with different numbers of data segments. The ratios in $\max(T_i)$ is computed relative to serial RJMCMC (283.346sec).

duced by almost half due to less computation in the shorter data chunks. Most of the tasks can complete in around 2.5sec, while tasks 15 and 16 possess the longest processing time, dominating the parallel performance. These two tasks correspond to chain_7 and chain_8 in the second data segment, in accordance with the longest chain in SSD-RJMCMC. It implies a possible range of k and the related peak positions. Under this setting, although there still exists load imbalance, this has been considerably reduced compared with SSD-RJMCMC.

If we set $s = 6$ and $s = 8$, both the longest convergence length and execution time are much reduced. In particular, compared with SSD-RJMCMC, the longest chain is shortened by half, and the largest timing difference of individual tasks is approximately decreased from 15sec to 2sec. The smaller and similar sized tasks are ideal to diminish load imbalance and reasonably support efficient parallel processing.

In conclusion, compared with standard serial RJMCMC on the real data set, the model-level parallelization in SSD-RJMCMC not only reduces the convergence length and the execution time of the concurrent chains, but also dramatically reduces the total amount of computations and total processing time. This provides the possibility to achieve a super-linear real speedup. However, it suffers from severe load imbalance. By introducing data-level parallelization, although DP SSD-RJMCMC$^{\text{U}}$ generates longer task queues and might slightly increase the total computation compared with SSD-RJMCMC, it further

reduces the individual task size, especially when the data chunks are reasonably short such that the peak returns are distributed among different histogram segments. The smaller and similar task sizes establish the precondition to solve the load balancing problem in SSD-RJMCMC, in assistance with the automatic scheduler for dynamic task allocation.

5.4.3 Statistical Property: Parameter Estimation Accuracy

5.4.3.1 RJMCMC Analysis of Incomplete Peak Return

In DP SSD-RJMCMC^{U}, since the LaDAR histogram is divided into chunks, we might observe some peaks spreading over two adjacent data segments. This experiment is designed to assess the ability of RJMCMC algorithm to detect the peak and estimate the peak position from the incomplete signals. Considering the analysis may be related to the proportion of peak return in each data segment, we slide a data boundary over the histogram, with a step spacing of 10 bins, as illustrated in Figure 5.11. To provide a benchmark of the peak position against which we place the data partitioning lines, we use the synthetic LaDAR response containing one real reflection of the known, measured instrumental response, with β and B both equal to 1. When shifting the division line from the right to the left to the peak maximum, the peak position relative to the data boundary changes gradually from negative values to positive values. The allowed variation for k is $[0, 2]$.

Figure 5.12 shows the estimation of relative peak position. Due to the symmetry of the instrumental response (see Figure 2.5(b)), the left and right data segments perform similarly in peak detection. Take the left one for example: First, when more than half of the peak return falls within the segment (Figure 5.11(a),(b) and (c)), RJMCMC can successfully and accurately capture the peak. Second, if the peak maximum overlaps the data boundary (Figure 5.11(d)), or falls outside the boundary by an extremely short distance (10 bins

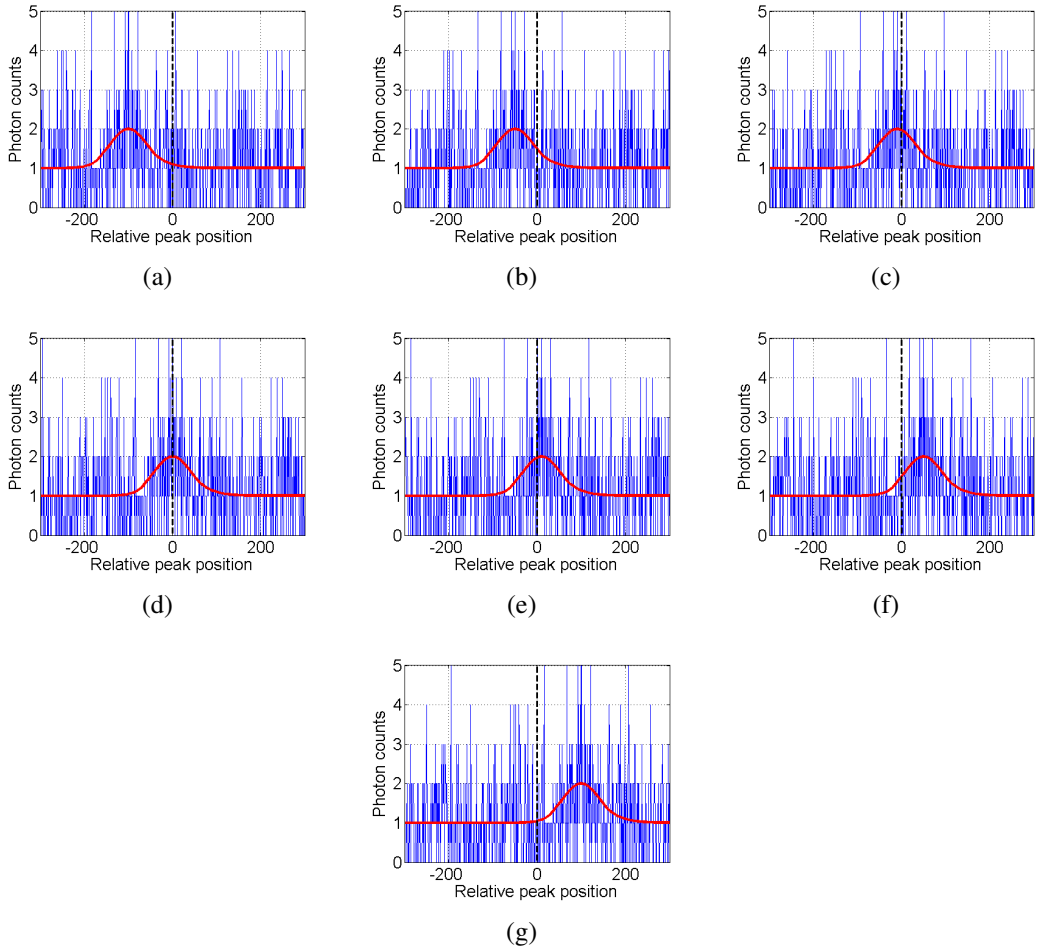


Figure 5.11: Data partitions slide over the peak return. (a) The left segment has the complete peak. (b,c,d,e,f) The peak return steps across two segments. (g) The peak return completely enters the right segment.

in Figure 5.11(e)), RJMCMC can still detect the peak but locates it inside the segment, since the valid proposal range for t_0 is within the semi-histogram. Accordingly, peak amplitudes are smaller than the ground truth to compensate the biased position. Third, when most of the peak enters the right segment (Figure 5.11(f)), RJMCMC fails to reconstruct the instrumental response from the short remaining tail. Although the sampler can still propose a peak and interpret the signal as the one within the segment, it gives a comparatively large error in t_0. This decreases the likelihood value and therefore causes a rejection of the proposal. For the signal segment, we would expect it could contribute the

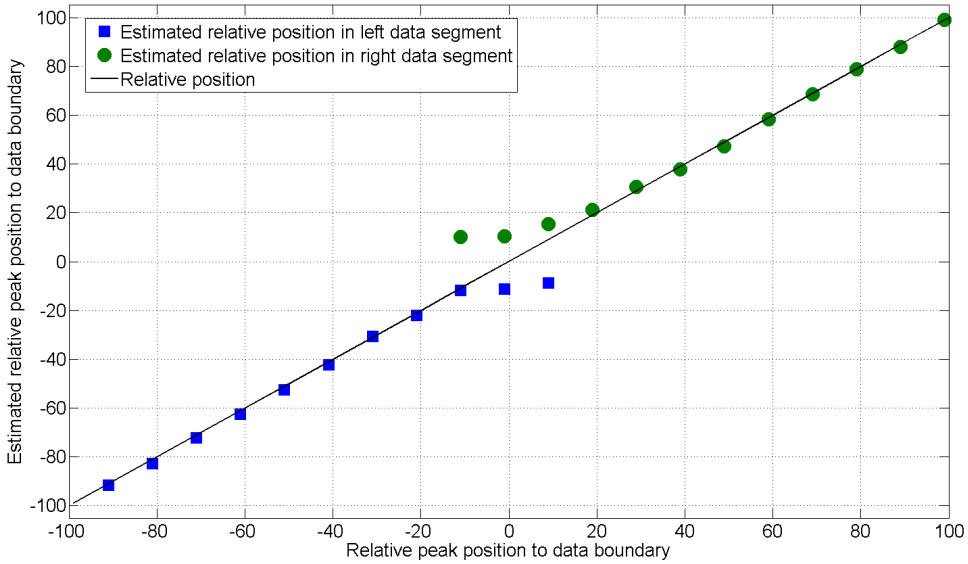


Figure 5.12: Estimated peak position for incomplete data using RJMCMC.

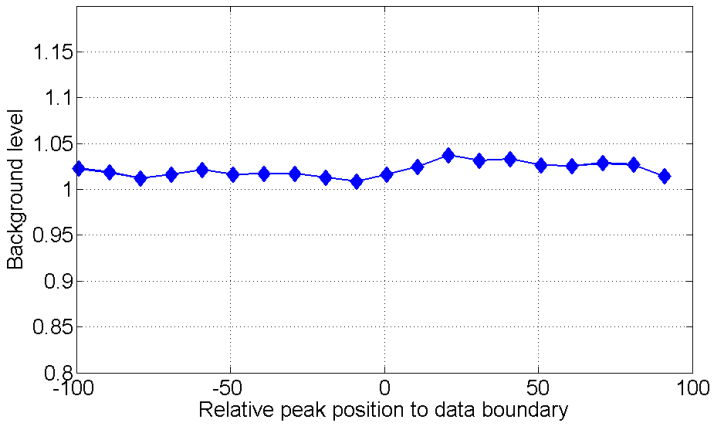


Figure 5.13: Estimated background level for incomplete data.

background level, but according to Figure5.13, B is not levelled up. Intrinsically, RJMCMC intends to fit the entire data segment rather than highlighting local "extraordinary" photon counts.

In summary, regarding the 1.7cm resolution (approximately 28 bins with 4ps time resolu-

tion) of the operational model concluded in [22], the estimation accuracy and sensitivity for incomplete data is still reasonably high, even though the reflection could be detected twice when it locates within 10 bins of the data boundary. The "smaller false return" is the price to be paid for the task queue generation and the potential speedup.

5.4.3.2 Synthetic Data

Figure 5.14 displays the box plot of positions. These plots allow us to detect and illustrate locations and variations of different peaks. It shows that the median values of the estimates tend to approach the true positions, and dispersions are acceptably small. The interquartile ranges are relatively small, suggesting accurate and stable estimations in DP SSD-RJMCMC^{U}. The final processing result from DP SSD-RJMCMC^{U} is shown in Figure 5.15.

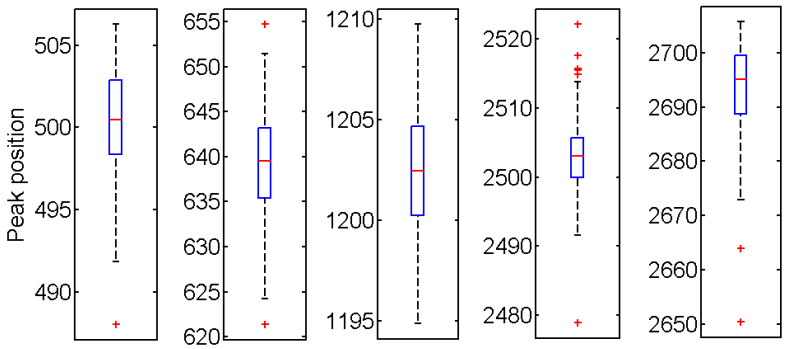


Figure 5.14: Boxplot of t_0 estimates for the synthetic data using DP SSD-RJMCMC^{U} algorithm.

5.4.3.3 Real Data

Figure 5.16 gives the box plot of positions in serial RJMCMC, SSD-RJMCMC and DP SSD-RJMCMC^{U}. For the real data, the peak number and positions are unknown. If we treat the estimates from standard RJMCMC as the benchmark, our developed algorithms

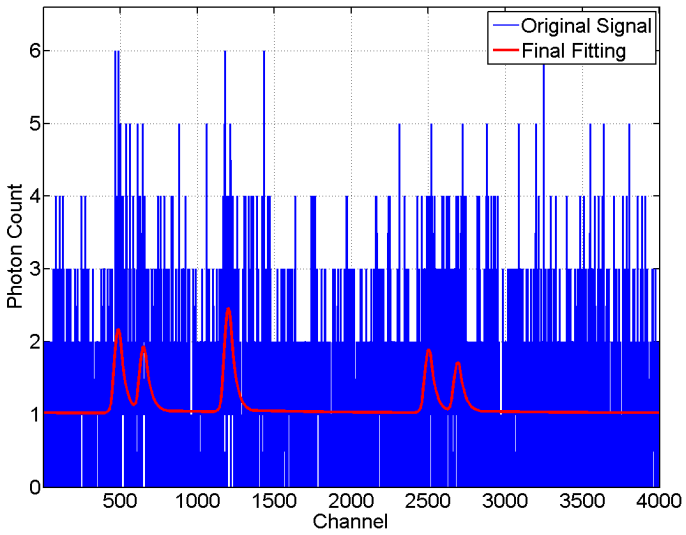
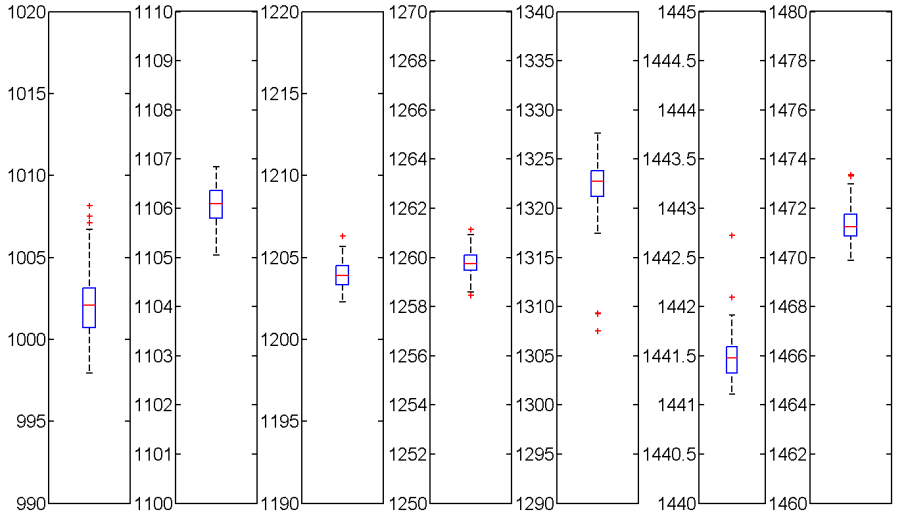


Figure 5.15: Final fitting result of the synthetic data using DP SSD-RJMCMC^{U} algorithm.

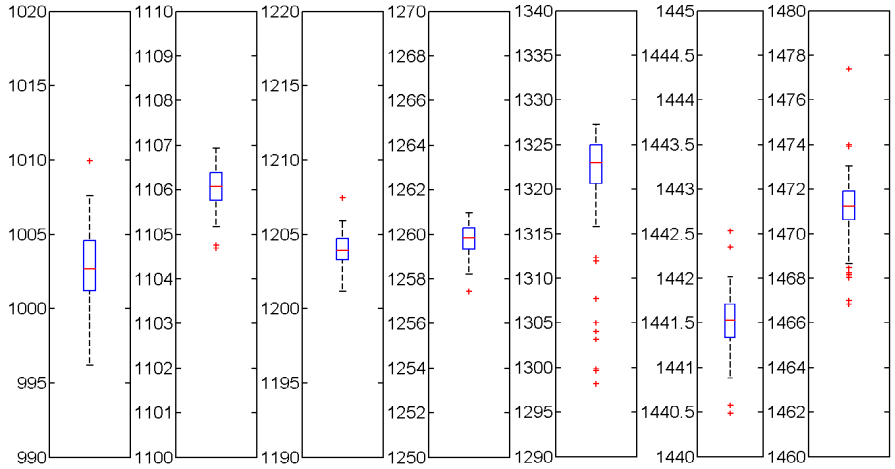
can successfully and correctly detect all the peak returns. Generally, the interquartile range is no more than 5 channel bins, equivalent to 0.6 mm in distance, a very high resolution for surface reconstruction. The small interquartile ranges also indicate sufficient parameter exploration for the determined model as well as the highly repeatable estimations. There are a few outliers in the plots, particularly the position around the 1320^{th} channel bin. This is caused by the analyzing difficulty in the target data due to the closely separated peaks and low peak amplitude. Figure 5.17 shows the final fitting results in DP SSD-RJMCMC^{U} and Figure 5.18 gives the 3D scatter plot for the scene in Figure 5.7.

5.4.4 MPI Implementation: Communication Efficiency

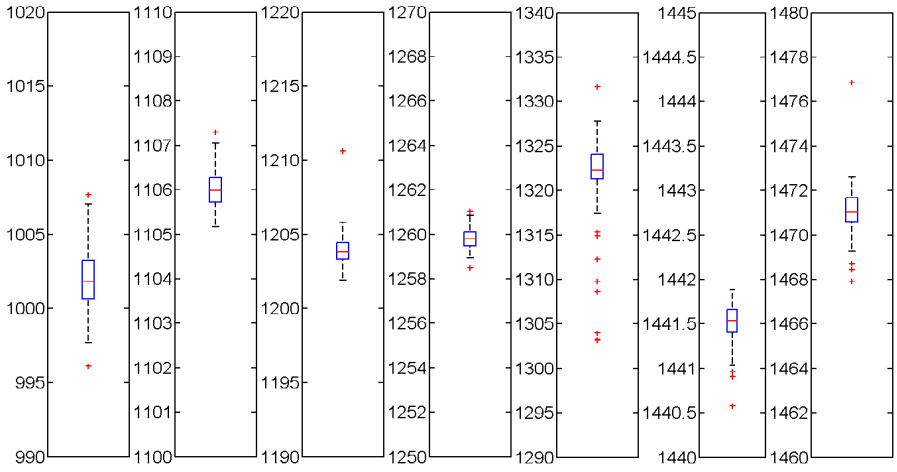
This section will evaluate the algorithm in the computing domain. We have demonstrated that with the data decomposition technique, DP SSD-RJMCMC^{U} further divides the tasks into numerous smaller ones with similar size. This directly helps to address the load balancing problem in SSD-RJMCMC, and meanwhile makes the algorithm adaptive to



(a)

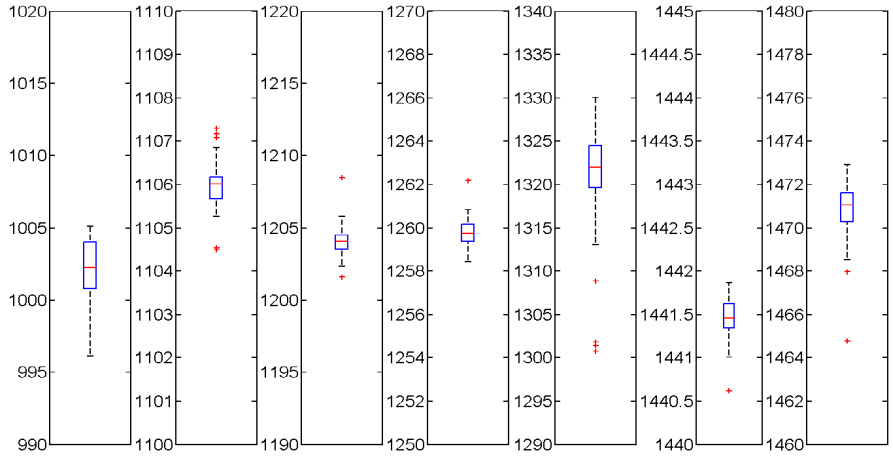


(b)

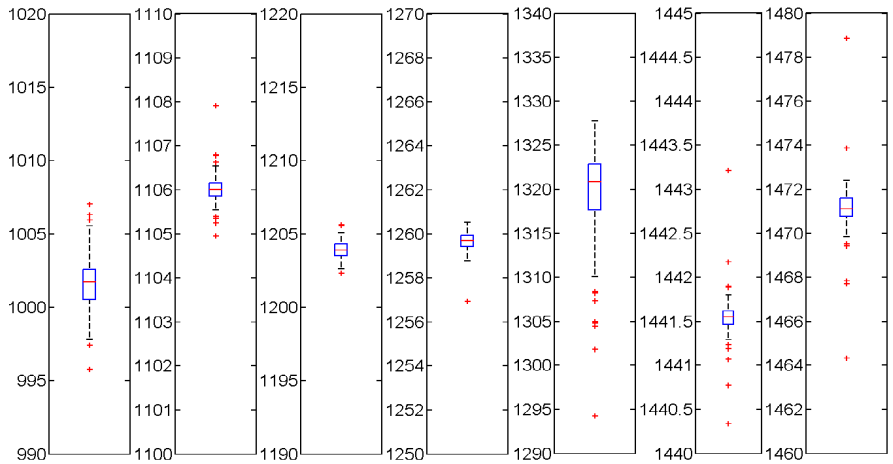


(c)

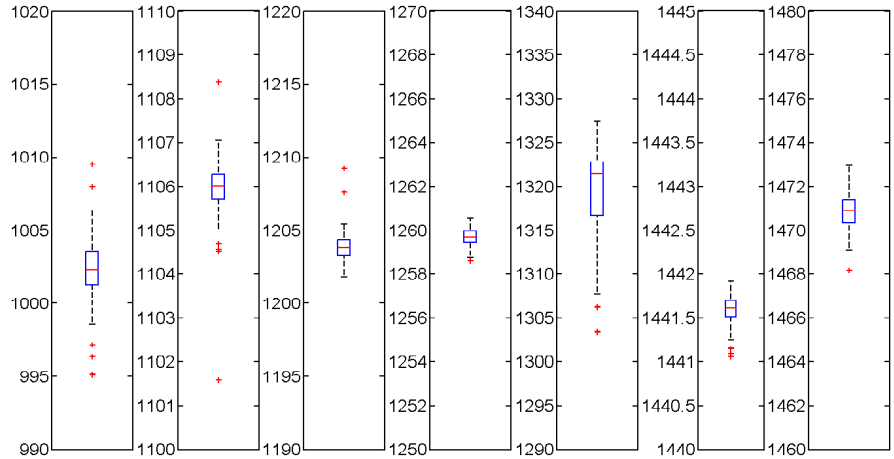
Figure 5.16: Box plot of t_0 estimates in RJMCMC (a), SSD-RJMCMC (b) and DP SSD-RJMCMC$^{\mathrm{U}}$ with $s = 2$ (c).



(d)



(e)



(f)

Figure 5.16: (Continued) Box plot of t_0 estimates in DP SSD-RJMCMC$^{\mathrm{U}}$ with $s = 4$ (d), $s = 6$ (e) and $s = 8$ (f).

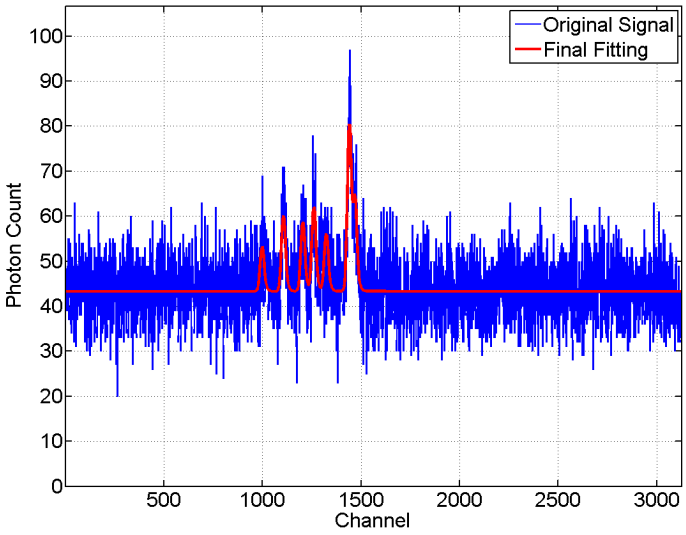


Figure 5.17: Final fitting result of the real data (shown in Figure 5.8(a)) using DP SSD-RJMCMC^{U}.

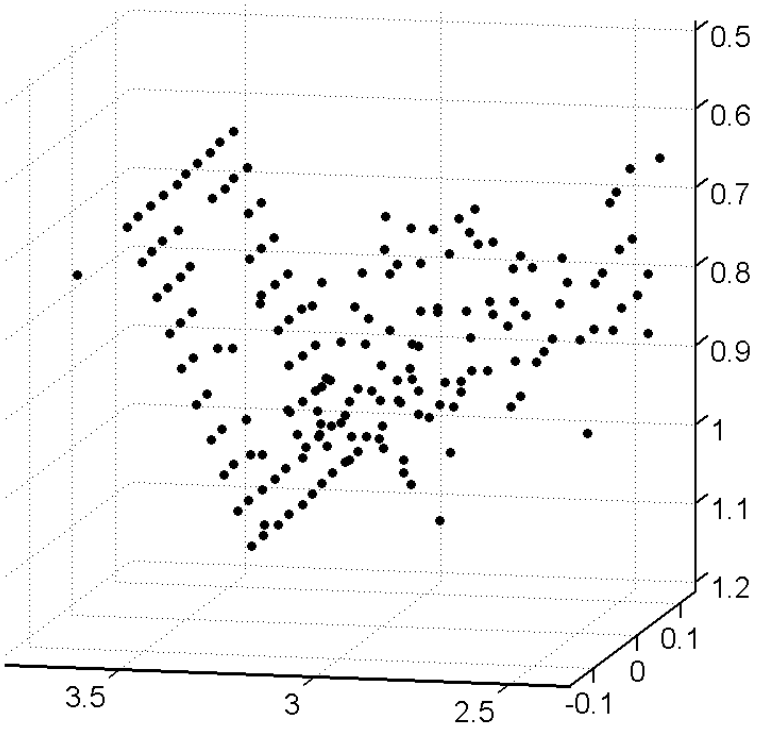


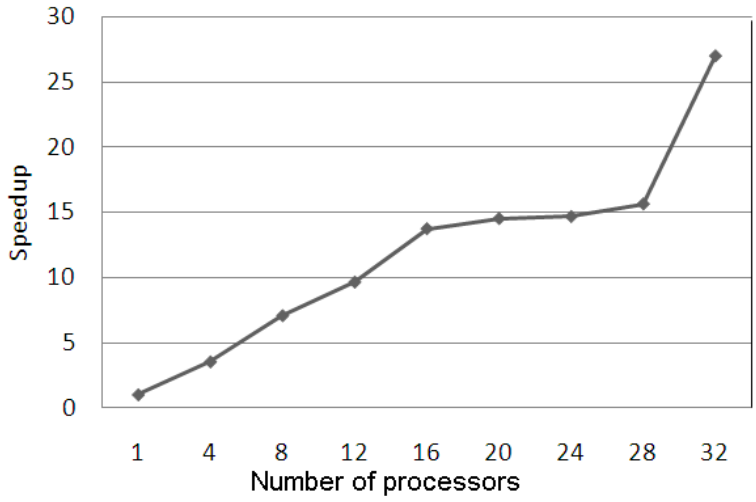
Figure 5.18: 3D scatter plot of the real target (in metres) presented in Figure 5.7 using DP SSD-RJMCMC^{U} algorithm.

different number of processors. Although these two main challenges stated in Section 5.2.3 can be successfully conquered, with reference to Section 3.3.1, the communication overhead is another critical issue directly determining the parallel performance. The following discussion will focus on the evaluation of communications in DP SSD-RJMCMC$^{\text{U}}$ using the synthetic data.

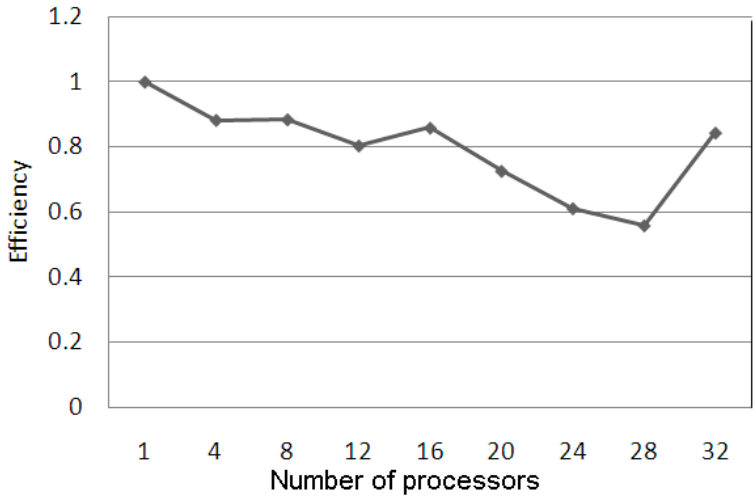
To assess the utilization of the processor, we filter out the performance difference caused by the random task size, generating 32 sequences with a fixed chain length of 300 samples. Figure 5.19 shows that when the processor number changes from 1 to 16, speedup increases greatly and efficiency stables around 0.85, which is very close to the theoretical threshold 1. The two reasons for the high level efficiency and the nearly linear speedup are as follows:

First, speedup is determined by the serial part of the programme. The smaller the proportion of serial computation, the larger the speedup. Since DP SSD-RJMCMC$^{\text{U}}$ parallelizes most of the serial computations used for sample generations, it dramatically reduces the time for serial processing.

Second, the execution time for each individual task is almost the same, from 0.414 to 0.531 seconds. This indicates that the load imbalance is almost eliminated, and the main factor affecting the implementation efficiency is the inter-note communication. As discussed in Section 5.3.3, the master and slaves have coarse granularity and communicate at a very low frequency, transmitting only $2 \times (32 - N_{\text{master}}) + 1$ messages in total, where N_{master} is the number of tasks executed by the master. To give a comparable result for the fine-grain task size, we parallelize the likelihood calculation and transfer the intermediate results for each sweep in RJMCMC. Figure 5.20 shows that the speedup improvement is almost negligible, and the implementation efficiency even degrades with 8 processors. This is because the frequent message passing almost sacrifices all the benefit from the parallel processing.



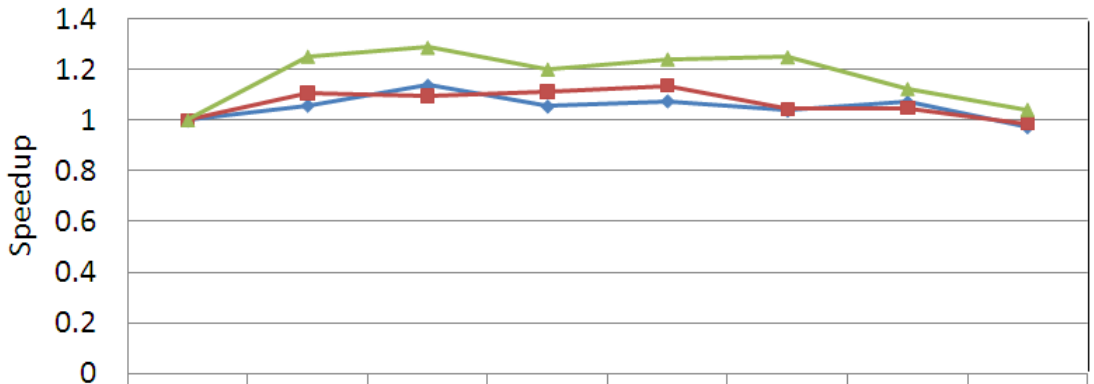
(a)



(b)

Figure 5.19: Speedup and efficiency of DP SSD-RJMCMC^{U} when working under the fixed convergence length.

In Figure 5.19, we also noticed that as the number of processors increases from 16 to 28, the speedup rises slowly. In this case, each processor is assigned with either one or two tasks with very similar execution time, which makes lack of synchronization problem a bottle neck for processor utilization. Even though more processors are employed, the total execution time is not shortened, until all the 32 nodes participate in and the loads are



	1	2	3	4	5	6	7	8
k35	1	1.05843	1.13713	1.05503	1.07575	1.0389	1.07224	0.97389
k46	1	1.10667	1.09624	1.11199	1.13631	1.04393	1.04809	0.98554
k57	1	1.25147	1.28671	1.20103	1.24059	1.24871	1.12404	1.04159

Figure 5.20: Speedup in triple-state RJMCMC chains when parallelizing the likelihood computation.

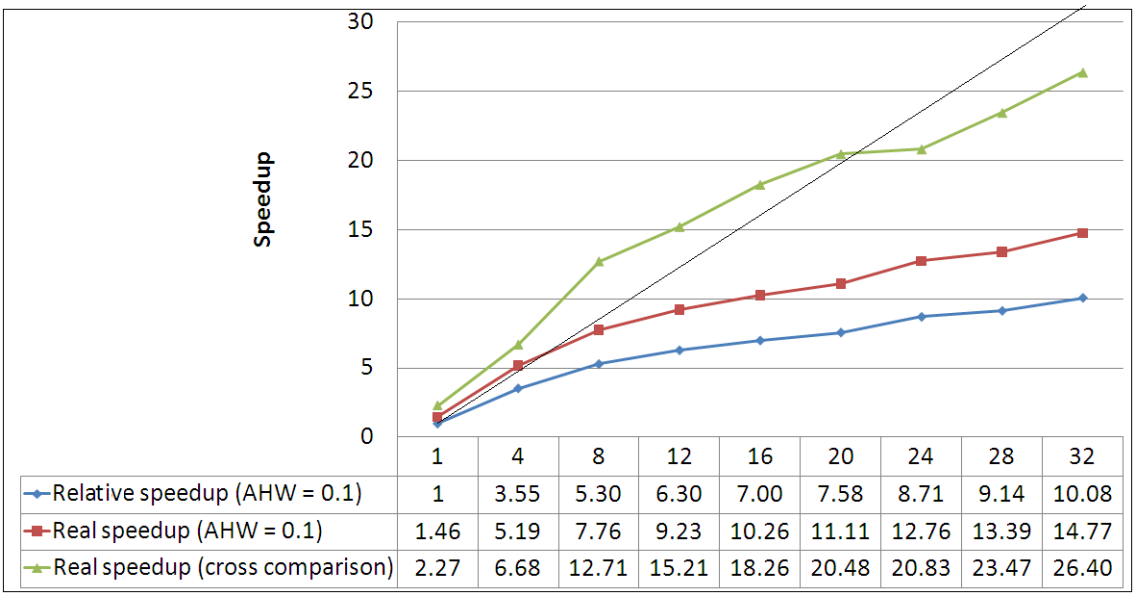
balanced again.

5.4.5 Speedup Achievement

This section presents the speedup achievements on both data sets. Since the problem size and the computation complexity in DP SSD-RJMCMC^{U} is not exactly the same as the serial RJMCMC algorithm, we compute both the *relative speedup* and the *real speedup* defined in Section 3.2.4. To evaluate the utilizations of the processors regarding the communication efficiency and the load balancing, we will analyze the Efficiency(p) using different number of processors (p).

5.4.5.1 Synthetic Data

Figure 5.21 displays the speedup for DP SSD-RJMCMC^{U} with dynamic chain length control. The *real* speedup is computed relative to the processing time of serial RJMCMC running on a single Beowulf machine. Under the same convergence condition (AHW $= 0.1$), we almost achieve a linear speedup of 7.8 with 8 processors. The speedup reaches 15 for 32 processors. Due to the load imbalance, this is not as high as in the fixed length scenario. When the number of processors is less than the task number, task mixing on separate processors compensate the difference of execution time. By contrast, load imbalance, although largely diminished by data parallelism, becomes obvious when generating all the 32 tasks concurrently.



	1	4	8	12	16	20	24	28	32
Relative speedup (AHW = 0.1)	1	3.55	5.30	6.30	7.00	7.58	8.71	9.14	10.08
Real speedup (AHW = 0.1)	1.46	5.19	7.76	9.23	10.26	11.11	12.76	13.39	14.77
Real speedup (cross comparison)	2.27	6.68	12.71	15.21	18.26	20.48	20.83	23.47	26.40

Figure 5.21: Real speedup and relative speedup of DP SSD-RJMCMC^{U} algorithms when AHW $= 0.1$, and the real speedup when AHW for RJMCMC and DP SSD-RJMCMC^{U} are 0.1 and 0.15 respectively.

The *relative* speedup uses the total execution time of the DP SSD-RJMCMC^{U} algorithm on a single processor. It is smaller than the real speedup since with reference to Table 5.3, serial processing of DP SSD-RJMCMC^{U} is slightly slower than RJMCMC for this syn-

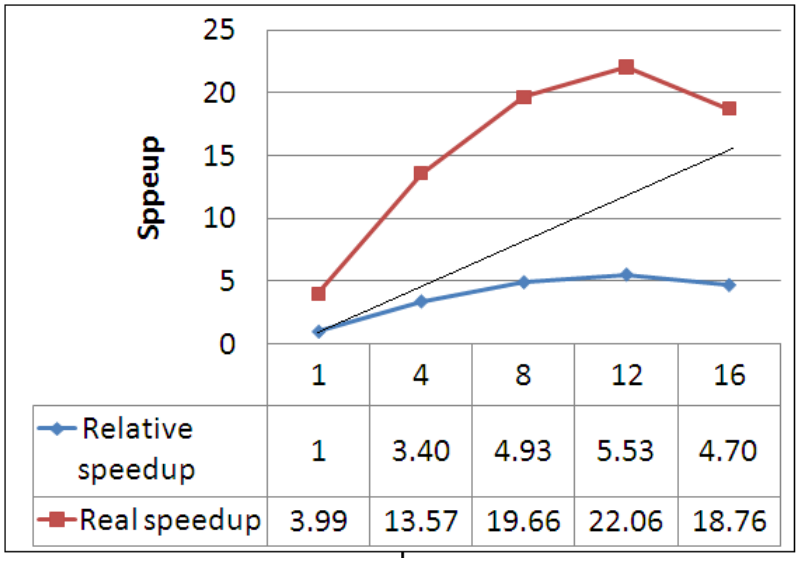
thetic data. Recalling Table 5.3, we achieve the best efficiency enhancement and meanwhile maintain the same estimation accuracy when DP SSD-RJMCMC^{U} works under a relaxed convergence condition, AHW $= 0.15$. This gives a speedup of 26.4, which is very close to the ideal case (speedup $= 27$) where the chain lengths are fixed and the load imbalance is diminished to the largest extend. When the processor number is no more than 20, we can achieve the super linear speedup, for example 12.7 with 8 processors, due to the further reduced computation.

5.4.5.2 Real Data

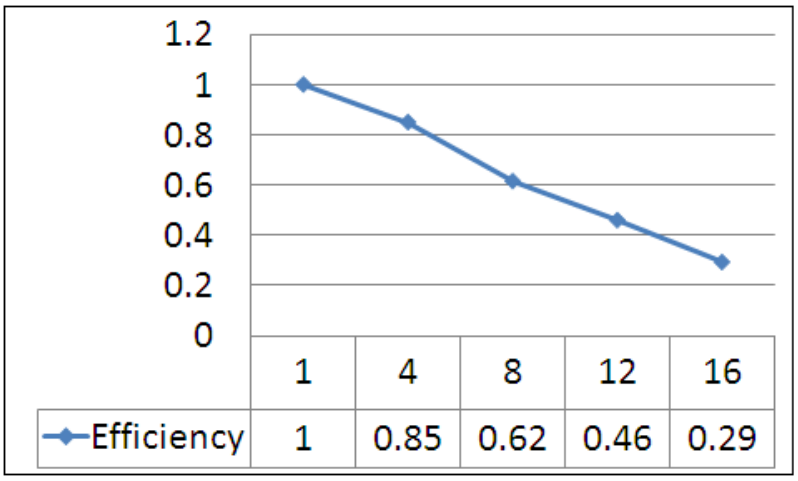
Figure 5.22 to Figure 5.25 display the relative/real speedup and the efficiency varying with different numbers of processors under different data parallelization conditions. We observed the super linear real speedup for this data set, since with reference to Table 5.5, the serial processing time of DP SSD-RJMCMC^{U} is up to 4.8 times less than serial RJMCMC. In particular, when $s = 6$, we achieved a speedup of 60.19 with 30 processors. This is in accordance with the distribution of N_i and T_i shown in Figure 5.10, which has the smallest and most similar task sizes, and hence achieves the best load balancing. This verifies that the speedup achievement in DP SSD-RJMCMC^{U} should be a composite effect of both of the improved sampling efficiency in the statistical domain, and the implementation efficiency benefit from the computing domain.

For this data set, DP SSD-RJMCMC^{U} achieves the highest relative speedup with 8 segments and 32 processors. For this configuration, Table 5.6 summarizes the timing results for each task executed on different machines. We noticed that compared with the longest task execution time (3.668863sec in task 14) for this random trial, the summation of the communication time (0.02sec) is very short. In this case, the main factor affecting the relative speedup is the load imbalance. For example, the master takes about 0.8sec to finish task 1, but after that, it has to wait for about 2.9sec to receive the processing result of task

14 from the worker. However, when we reduce the number of processors to 8, some of the machines will execute one to two larger sized tasks, while the others can be assigned with more smaller sized tasks. The mixture of the different sized tasks on each individual machine helps to balance the load. This explains why the relative speedup tails off around 8 processors.



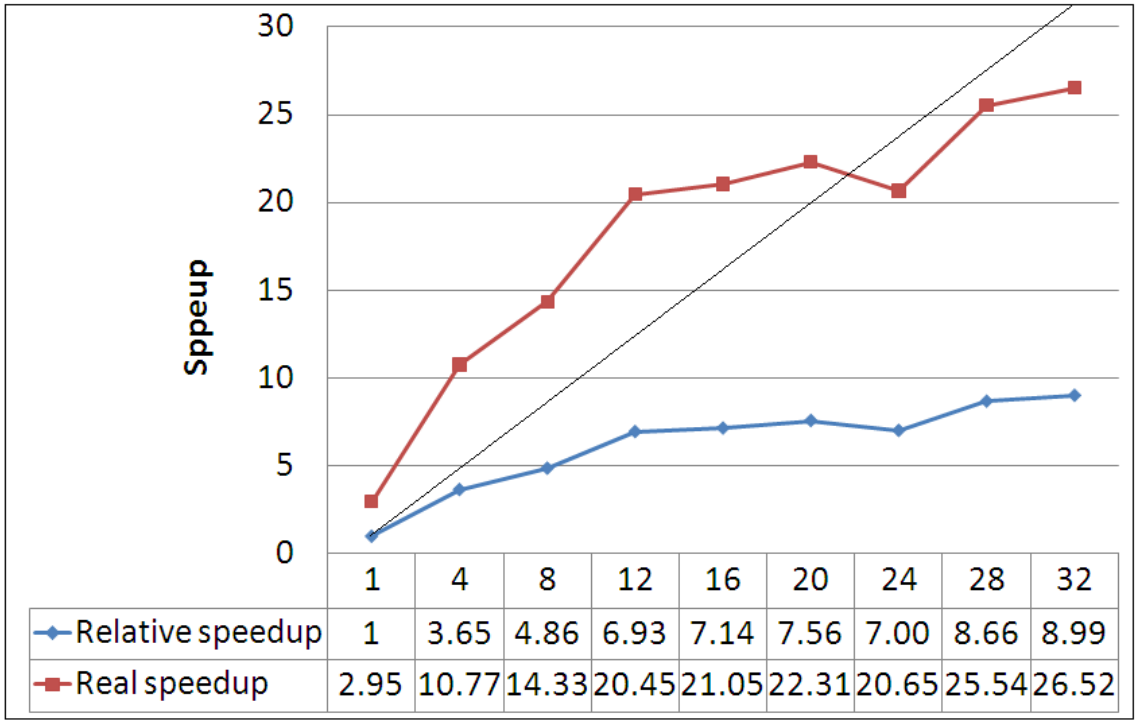
(a)



(b)

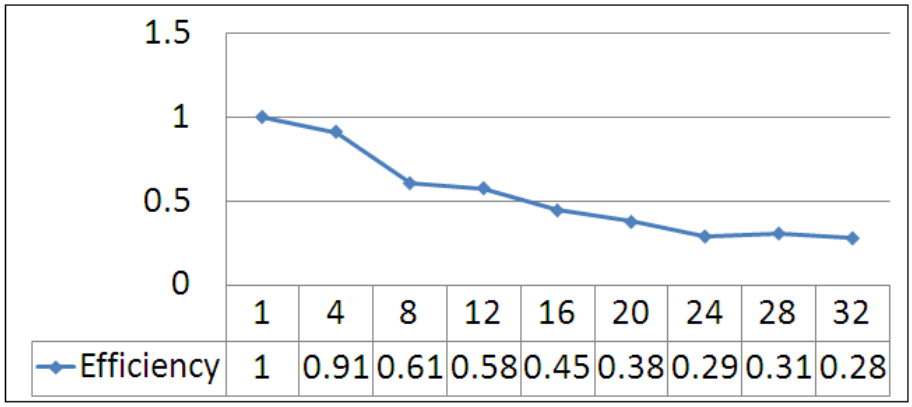
Figure 5.22: Speedup(p) and efficiency(p) in DP SSD-RJMCMC$^{\mathrm{U}}$ with $s = 2$.

Generally, efficiency(p) decreases as the number processors p increases. As discussed in



	1	4	8	12	16	20	24	28	32
Relative speedup	1	3.65	4.86	6.93	7.14	7.56	7.00	8.66	8.99
Real speedup	2.95	10.77	14.33	20.45	21.05	22.31	20.65	25.54	26.52

(a)

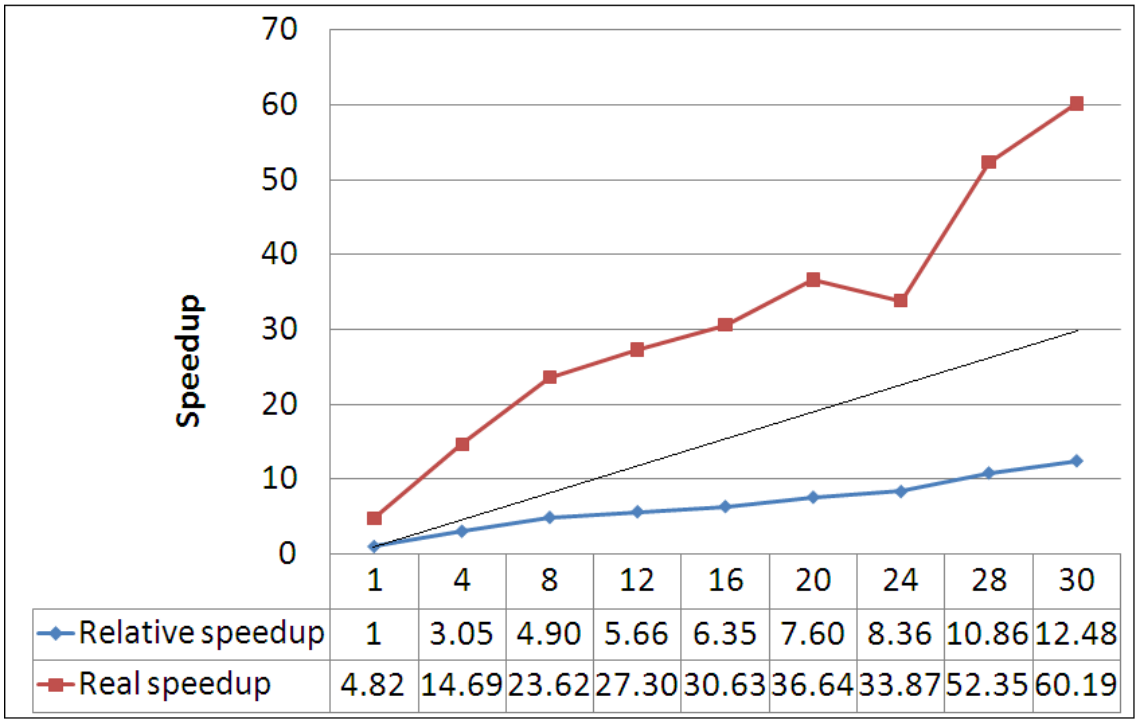


	1	4	8	12	16	20	24	28	32
Efficiency	1	0.91	0.61	0.58	0.45	0.38	0.29	0.31	0.28

(b)

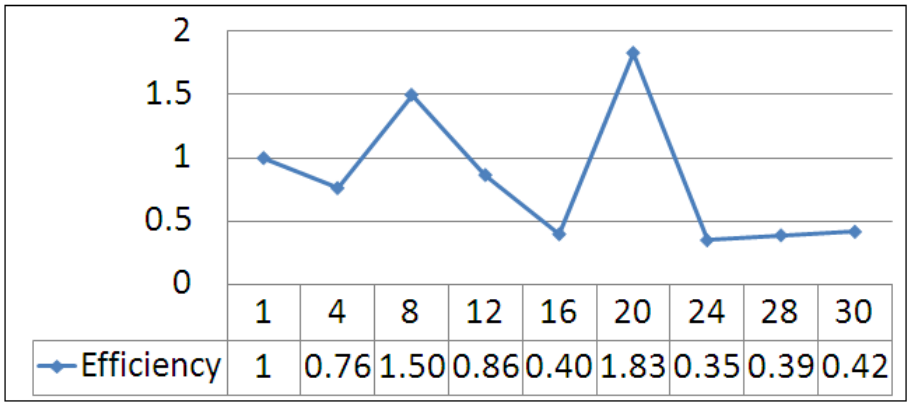
Figure 5.23: Speedup(p) and efficiency(p) in DP SSD-RJMCMC$^{\mathrm{U}}$ with $s = 4$.

Section 5.3.3, when we include more nodes, we introduce more inter-processor communications. The other reason is that the automatic scheduler for dynamic task allocation can mix the different sized tasks better and diminish the load imbalance to the largest extent when the number of processors is much smaller than the total number of tasks.



	1	4	8	12	16	20	24	28	30
Relative speedup	1	3.05	4.90	5.66	6.35	7.60	8.36	10.86	12.48
Real speedup	4.82	14.69	23.62	27.30	30.63	36.64	33.87	52.35	60.19

(a)

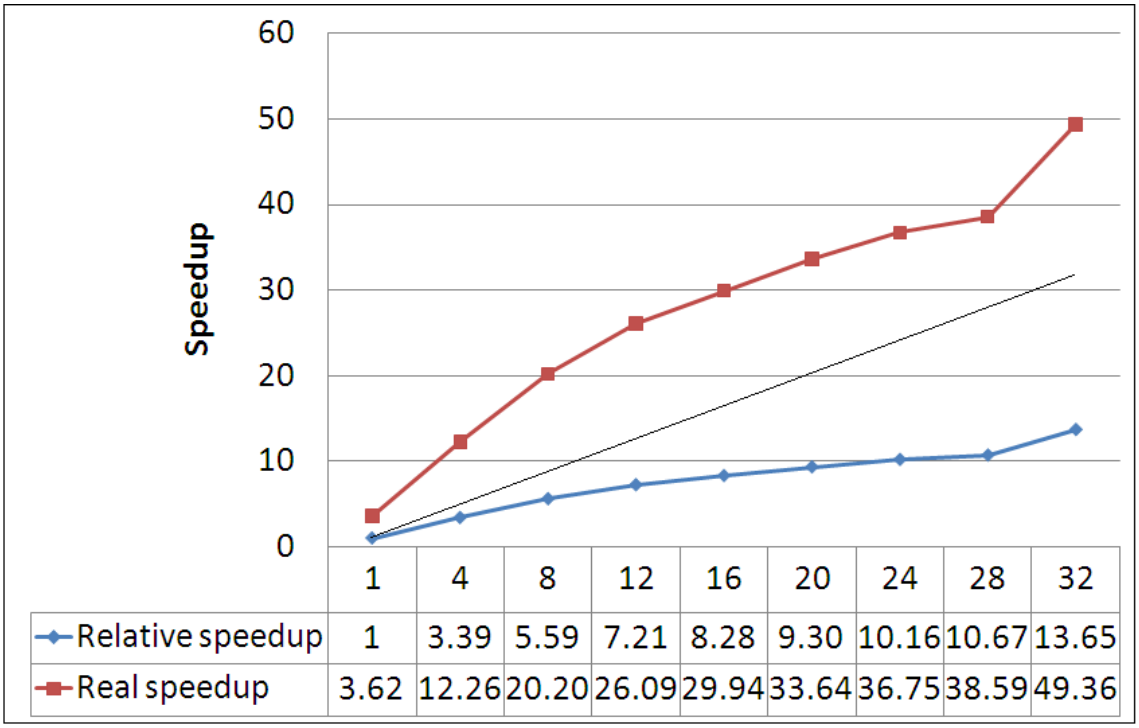


	1	4	8	12	16	20	24	28	30
Efficiency	1	0.76	1.50	0.86	0.40	1.83	0.35	0.39	0.42

(b)

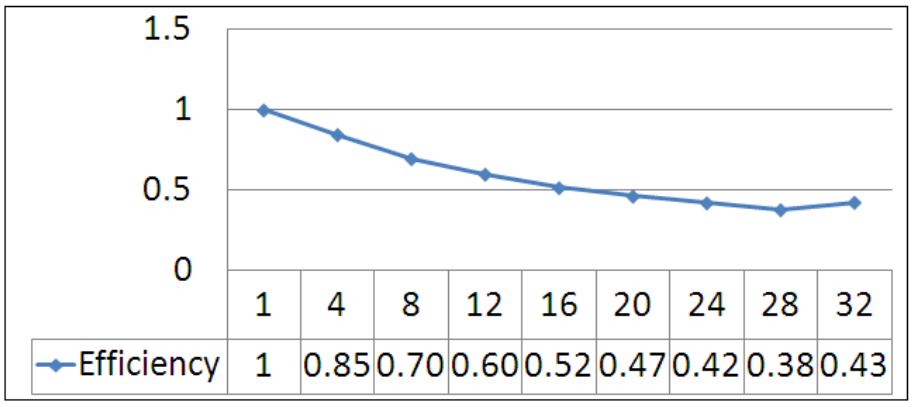
Figure 5.24: Speedup(p) and efficiency(p) in DP SSD-RJMCMC$^{\text{U}}$ with $s = 6$.

When these two numbers become comparable, speedup is flat or has small fluctuations. For instance, for $s = 2$, the speedup degrades when increasing from to 16. Therefore, we should be cautious about the trade-off between speedup(p) and the efficiency(p) when choosing the number of processors involved in parallel processing.



	1	4	8	12	16	20	24	28	32
Relative speedup	1	3.39	5.59	7.21	8.28	9.30	10.16	10.67	13.65
Real speedup	3.62	12.26	20.20	26.09	29.94	33.64	36.75	38.59	49.36

(a)



	1	4	8	12	16	20	24	28	32
Efficiency	1	0.85	0.70	0.60	0.52	0.47	0.42	0.38	0.43

(b)

Figure 5.25: Speedup(p) and efficiency(p) in DP SSD-RJMCMC$^{\mathrm{U}}$ with $s = 8$.

5.5 Conclusions

We have proposed a parallel implementation of SSD-RJMCMC algorithm for LaDAR signal analysis. In the proposed DP SSD-RJMCMC$^{\mathrm{U}}$ framework, we partition the photon

Task ID	Initialization	Task execution	Communication
1	0.007399	0.810336	0.000169
2	0.005879	1.712960	0.000047
3	0.005875	2.167007	0.000029
4	0.005918	1.792916	0.000309
5	0.005560	0.481678	0.000046
6	0.006072	2.634982	0.000318
7	0.006085	2.509846	0.000342
8	0.006091	2.499851	0.000312
9	0.005974	1.482708	0.000045
10	0.006062	1.840399	0.000372
11	0.006113	2.788105	0.000324
12	0.005729	1.925478	0.000311
13	0.005673	1.773685	0.000298
14	0.005882	3.668863	0.000038
15	0.005884	1.800926	0.000278
16	0.005867	1.828391	0.000316
17	0.006024	0.696894	0.000048
18	0.005972	1.861381	0.000306
19	0.005917	1.862254	0.000255
20	0.005974	1.769057	0.000261
21	0.006006	2.333002	0.000307
22	0.005901	1.903256	0.000033
23	0.005693	2.019496	0.000313
24	0.005874	2.303002	0.000294
25	0.005692	0.570398	0.000401
26	0.005940	1.853639	0.000042
27	0.005984	1.859836	0.000304
28	0.005968	1.775645	0.000030
29	0.005819	0.940761	0.000043
30	0.006064	1.682034	0.000021
31	0.005971	1.699986	0.008190
32	0.005990	1.989367	0.005863
All		max = 3.668863	0.02

Table 5.6: Timing (sec) of DP SSD-RJMCMC$^{\mathrm{U}}$ with $s = 8$, $p = 32$. Task 1 is executed on the master, and the other tasks are executed on separate processors simultaneously. For master only, the task execution time includes the chain generation and the tasks allocation, while the communication time is the extra time after the master finishes its own task, spent in receiving the processing results from the busy workers.

count histogram into separate segments, and apply data-level parallelization on top of the model-level parallelization. The two-dimensional task grid is transformed to a queue, and the tasks are dynamically assigned to different processors according to the designed scheduler.

Statistically, it inherits the strength of SSD-RJMCMC with an improved model selection accuracy compared with standard RJMCMC, and can further reduce the convergence length of the parallel sequences. In the computing domain, the tasks have coarse granularity and therefore the processors communicate at a very low rate. Additionally, it successfully addresses the load-balancing problem in SSD-RJMCMC, and makes the algorithm adaptive to different number of processors.

The designed framework was implemented on a 32-node Beowulf cluster using the MPI standard. First, we have shown the capability of RJMCMC to interpret an incomplete LaDAR response. Although a surface can be detected twice if located extremely close to the data partition, the algorithm maintains high level estimation accuracy and sensitivity relative to the system resolution. Second, for the statistical property, the DP SSD-RJMCMC^{U} can further improve the sampling efficiency by reducing the convergence length and computation time for each formalized task. Third, for the efficiency of parallel implementation, when evaluated with the fixed Markov chain length on the synthetic data, DP SSD-RJMCMC^{U} can achieve a parallel efficiency as high as 0.85, a speedup of 7 with 8 processors and 27 with 32 processors, which demonstrates the effectiveness of the diminished load imbalance and infrequent communication. In comparison, due to the massive message passing in the parallel likelihood method, we gain almost no speedup or even rise in execution time.

If we apply the dynamic chain length control, for the synthetic data, we can achieve a nearly linear speedup of 7.8 with 8 processors relative to RJMCMC under the same convergence condition. If we relax the convergence condition for DP SSD-RJMCMC^{U},

we reach a speedup of 26.4 with 32 processors, and a super-linear speedup of 12.7 with 8 processors, but with the retained estimation accuracy. For the real data set, we can achieve super linear speedup under several conditions, for example, a speedup 20.5 with 12 processors for $s = 4$, 60.2 with 30 processors for $s = 6$, 50.3 with 32 processors for $s = 8$ and etc.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

The work described in this thesis has been concerned with Bayesian analysis of full waveform LaDAR data obtained from TCSPC laser systems. In particular, Bayesian analysis using RJMCMC has outstanding strengths in resolving multiple surface returns within the laser footprint, and creating multilayer images. We have demonstrated the application of RJMCMC to analyze full waveform LaDAR pixels and image data acquired by our scanning sensor. The comparisons with conventional correlation analysis highlight the advantages of RJMCMC in detecting low amplitude returns and reconstructing closely separated surfaces.

Despite its high resolution and sensitivity, RJMCMC is computationally expensive. Hence, a major goal of this thesis was to improve the simulation efficiency of RJMCMC using parallel computing on a distributed memory cluster using MPI programming. This was not a direct data or control parallel implementation. Statistically, parallelizing the serial sampling procedure in RJMCMC and retaining the parameter estimation accuracy were

significant challenges.

1. We evaluated the sampling efficiency by adaptive analysis of convergence and so dynamically terminated the chain generation. This was challenging because the chain length is not consistent from execution to execution, again making simple parallel implementation more difficult. We implemented the Gelman and Rubin and Heidelberger and Welch diagnostics (HWD). Both can effectively monitor the mixing performance of the Markov chain and successfully conclude a convergence. Especially, HWD can automatically detect and remove the burn-in period on-the-fly, which frees us from running training sequences to manually determine a safe burn-in length for all the random trials.

2. We also implemented a strategy of parallel likelihood computation. Previously applied to parallel MCMC algorithms, it need not be restricted to the fixed dimensional problem, and can be directly applied to RJMCMC. However, application to the coal mining disaster problem, a benchmark problem for varying-dimensional signal analysis, and LaDAR problems provided no discernible benefit, due to the frequent message passing. This motivated us to develop a parallel strategy with coarse granularity and less communication.

3. We then implemented parallel MCMC chains method for varying-dimensional signal processing. Applying these to the coal mining disaster problem demonstrated that the within-model parameter updates in MCMC have better mixing performance than between-model jumps in RJMCMC, and hence converge faster with shorter lengths. In addition, because of the independence of MCMC chains exploring models with different dimensionality, there is no inter-processors communication during the sampling procedure, which improves speedup. However, for LaDAR signals containing multiple surface returns, the MCMC sampler has poor convergence. Some parameters do not escape from the burn-in period and settle down to a covariant stationary process, even with the additional delayed rejection step.

Therefore, we needed to develop a valid RJMCMC parallelism approach to interpret LaDAR data.

4. We proposed an effective, concurrent RJMCMC algorithm, SSD-RJMCMC, which divided the entire state space into groups, and within each group generated an independent triple-state RJMCMC chain with restricted variation of model dimensions. Applying HWD, we achieved dynamic chain length control over the generated chains. This framework intrinsically has a parallel structure built on state space decomposition and reconfiguration, a form of model-level parallelization. Application to both synthetic and real data demonstrated that SSD-RJMCMC inherits the benefits of serial RJMCMC and parallel MCMC chains methods but addresses their problems. First, compared with serial RJMCMC, the concurrent triple-state RJMCMC chains have shorter convergence lengths and hence achieve higher simulation efficiency. Second, borrowing the idea of parallel exploration of the candidate models, it effectively solves the local optimal problem in RJMCMC. Third, in conjunction with the error detection and correction scheme, SSD-RJMCMC has lower model selection error and improves the reliability of peak detection. Fourth, based on the convergence assessment of the parameters, we were able to adaptively generate a complimentary MCMC chain to more rapidly refine the within-model parameter updates and ensure the accuracy of final estimation. However, considering the fixed number of parallel chains with very different convergence lengths, this algorithm still had difficulty in achieving load-balancing on a parallel platform, and was not easily adapted to different numbers of processors.

5. For this reason, we employed SSD-RJMCMC as a statistical prototype and developed a degree of data parallelism to improve load-balancing, DP SSD-RJMCMC$^{\mathrm{U}}$. The evaluations on both synthetic and real data have demonstrated that DP SSD-RJMCMC^{U} retained the statistical advantages of SSD-RJMCMC but overcomes its two shortcomings. By adding data-level on top of the model-level parallelization, it formalizes a task queue with smaller and similar sized tasks. The automatic sched-

uler dynamically allocates waiting tasks onto idle processors. These two strategies successfully diminish the load imbalance that occurred in SSD-RJMCMC. Moreover, due to coarse granularity and the independence between tasks, there is no message during the task execution. As a variant of RJMCMC, DP SSD-RJMCMC^{U} can reduce the problem size and computational complexity. Therefore, it can achieve a form of super linear speedup if the number of data segments and processors are chosen wisely.

6.2 Future Works

This thesis confirms that the RJMCMC approach can not only provide 3D data of higher resolution and detect weak signatures than the deterministic techniques, but also reconstruct multiple layer images where the reflected signal diverges to hit the multiple surfaces, or is multiply reflected from semi-transparent objects. Coupled with advances in detector and system design, the parallel algorithms we have developed offer the prospect of real-time dense 3D multi-layer image capture. However, our current Bayesian inference of LaDAR signals on Beowulf cluster takes processing times in seconds. In the long term, we need to process the data at a comparable rate to sub-second data acquisition. Both algorithmic and hardware improvements are required in order to permit efficient implementations in the near future.

First, we have implemented an efficient parallel RJMCMC on the distributed memory platform. Now, hybrid parallel systems, incorporating multi-core processors within each machine, are the emerging and inevitable trend. Based on the shared memory structure, the multi-core machines have faster access to data via a high speed data bus and avoid inter processors communications via network. Starting with DP SSD-RJMCMC^{U} algorithm, we would aim to combine the MPI implementation with OpenMP, parallelizing the

task execution on the multi-core processors within individual machines. This has a great potential to manipulate the processing behaviour of the implementation.

Second, we would aim to develop rapid and efficient embedded hardware. The DP SSD-RJMCMC^{U} algorithm involves time-varying processing, which means the processing time and memory requirements can vary significantly during execution. The algorithm complexity is data dependent and the convergence length of the Markov chains is random in different trials. Therefore, we should choose a suitable hardware platform which permits adaptive memory allocation and dynamic reconfiguration. Field Programmable Gate Arrays (FPGAs) are mature multi-processor platform. However, previous implementations have demonstrated that the system-level processing of RJMCMC for LaDAR is not efficient on an FPGA, since the FPGA is more applicable to static data algorithms rather than the random statistical model employed here. Having said that, there are expensive parts of the algorithm, notably likelihood computation, that could be implemented effectively on an FPGA, if this could communicate effectively with a host CPU. Another possibility is to use Graphic Processing Unit (GPU) systems. Their highly parallel structure may make them more suitable and effective than general-purpose FPGA, particularly in processing numerous coarse-grain tasks in parallel.

References

[1] P. J. Green and A. Mira. Delayed rejection in reversible jump Metropolis-Hastings. *Biometrika*, 88:1035–1053, 2001. xiii, 42

[2] `http://top500.org`. xvi, 86, 87

[3] C. Sun. Uncalibrated three-view image rectification. *Image and Vision Computing*, 21:259–269, 2003. 9

[4] D. Mery and M. Carrasco. Automated multiple view inspection based on uncalibrated image sequences. *SCIA 2005, LNCS 3540*, pages 1238–1247, 2005. 9

[5] J. Ens and P. Lawrence. An investigation of methods for determining depth from focus. *IEEE Trans. Pattern Anal. Mach. Intell.*, 15:97–108, 1993. 9

[6] P. Favaro and S. Soatto. A geometric approach to shape from defocus. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(3):406–417, 2005. 9

[7] R. Zhang, P. S. Tsai, J. E. Cryer, and M. Shah. Shape from shading: A survery. *IEEE Trans. Pattern Anal. Mach. Intell.*, 21(8):690–706, 1999. 9

[8] U. R. Dhond and J. K. Aggarwal. Structure from stereo: A review. *IEEE Trans. Systems, Man and Cybernetics*, 19(6):1489–1510, 1989. 9

[9] N. Uchida, T. Shibahara, T. Aoki, H. Nakajima, and K. Kobayashi. 3d face recognition using passive stereo vision. *IEEE International Conference on Image Processing*, 2005. 9

[10] M. C. Amann, T. Bosch, M. Lescure, R. Myllyla, and et al. Laser ranging: a critical review of usual techniques for distance measuremen. *Opt. Eng.*, 40(1):10–19, 2001. 10

[11] M. Hebert. Active and passive range sensing for robotics. *In Proc. of 2000 IEEE International Conference on Robotics and Automation*, 1:102–110, 2004. 10

[12] M. A. Albota, R. M. Heinrichs, D. G. Kocher, D. G. Fouche, and et al. Three-dimensional imaging laser radar with a photoncounting avalanche photodiode array and microchip laser. *Appl. Opt*, 41:7671–7678, 2002. 11

[13] C. Niclass, A. Rochas, P. A. Besse, and E. Charbon. Design and characterization of a CMOS 3-D image sensor based on single photon avalanche diodes. *IEEE J. Solid-State Circuits*, 40(9):1847–1854, September 2005. 11

[14] M. Browder, B. Evans, J. Beck, M. Blessinger, and et al. Three dimensional imaging sensors program. *Proc. SPIE*, 4377:73–83, 2001. 11

[15] K. Johnson, M. Vaidyanathan, S. Xue, W. Tennant, and et al. Adaptive LaDAR receiver for multispectral imaging. *Proc. SPIE*, 4377, 2001. 11

[16] C. Ho, K. L. Albright, A. W. Bird, J. Bradley, and et al. Demonstration of literal three-dimensional imaging. *Appl Opt.*, 28(9):1833–1840, 1999. 11

[17] B. L. Stann, A. Abou-Auf, K. Aliberti, and et al. Research progress on a focal plane array ladar system using chirped amplitude modulation. *In Proc. of SPIE. Laser Radar Technology and Applications VIII*, 5086, 2003. 12

[18] R. Schneider, P. Thrmel, and M. Stockmann. Distance measurement of moving objects by frequency modulated laser radar. *Opt. Eng*, 40(1):33–37, 2001. 12

[19] G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6:461–464, 1978. 16

[20] A. P. Dempster, N. M. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm (with discussion). *Journal of the Royal Statistical Society Series B-Methodological*, 39:1–38, 1977. 16

[21] C. Fraley and A. E. Raftery. Bayesian regularization for normal mixture estimation and model-based clustering. *Journal of Classification*, 24:155–181, 2007. 17

[22] S. Hernandez-Marin, A. M. Wallace, and G. J. Gibson. Bayesian analysis of Lidar signals with multiple returns. *IEEE Trans Pattern Anal Mach Intell*, 29(12):2170–2180, December 2007. 17, 70, 83, 194

[23] A. M. Wallace, R. C. W. Sung, G. S. Buller, and et al. Detecting and characterising returns in a pulsed ladar system. *IEE Proceedings - Vision, Image and Signal Processing*, 153(2):160–172, April 2006. 17

[24] U. Grenander and M. I. Miller. Representations of knowledge in complex systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, 56(4):549–603, 1994. 18

[25] M. Stephens. Bayesian analysis of mixture models with an unknown number of components - an alternative to reversible jump methods. *The Annals of Statistics*, 28(1):40 – 74, 2000. 18

[26] K. K. Berthelsen and J. Moller. Bayesian analysis of markov point processes in. In A. Baddeley, P. Gregori, J. Mateu, R. Stoica, and D. Stoyan, editors, *Case Studies in Spatial Point Process Modeling*, volume 185 of *Springer Lecture Notes in Statistics*, page 85 ?97. Oxford University Press, Oxford, 2003. 18

[27] J. Moller. Markov chain monte carlo and spatial point processes. In O. Barndorff-Nielsen, W. S. Kendall, and M. N. M. van Lieshout, editors, *Stochastic Geometry, Likelihood and Computation,*, Proceedings Seminaire Europeen de Statistique. Chapman and Hall, 1997. 18

[28] O. Cappe, C. P. Robert, and T. Ryden. Reversible jump MCMC converging to birth-and-death MCMC and more general continuous time samplers. *J. Roy. Statist. Soc. B*, 65(3):679?00, 2003. 19

[29] M. Unser. Splines: A perfect fit for signal and image processing. *IEEE Signal Processing Magazine*, 16(6):22–38, 1999. 19

[30] A. El-Baz and G. Gimel′frab. Em based approximation of empirical distribution with linear combinations of discrete Gaussians. *IEEE ICIP*, 2007. 19

[31] M. A. Hofton, J. B. Minster, and J. B. Blair. Decomposition of laser altimeter waveforms. *IEEE Trans. on Geoscience and Remote Sensing*, 38(4):1989 – 1996, 2000. 19

[32] C. Mallet F. Lafarge, M. Roux, U. Soergel, F. Bretar, and C. Heipke. A marked point process for modeling lidar waveforms. *IEEE Transactions on Image Processing*, 19(12):3204–3221, December 2010. 19

[33] S. Pellegrini, G. Buller, J. Smith, A. Wallace, and et al. Laser-based distance measurement using picosecond resolution TCSPC. *Meas. Sci. Technol.*, 11:712–716, 2000. 20

[34] C. P. Robert and G. Casella. *Monte Carlo statistical methods*. Springer texts in statistics, 1999. 27

[35] L. Tierney. Markov chains for exploring posterior distributions (with discussion). *Annals of Statistics*, 22:1701–1762, 1994. 28, 29

[36] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equations of state calculations by fast computing machines. *J. Chem. Phys.*, 21:1087–1091, 1953. 30

[37] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Trans. Pattern Anal. Mach. Intell*, 6:721–741, 1984. 30

[38] S. Chib and E. Greenberg. Understanding the Metropolis-Hastings algorithm. *American Statistician*, 49:327–335, 1995. 30

[39] P. J. Green. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 82:711–732, 1995. 33, 116

[40] P. J. Green. Trans-dimensional Markov chain Monte Carlo. In *Highly Structured Stochastic Systems*. Oxford University Press, 2003. 33, 122, 123, 134

[41] M. Sipser. Measuring complexity. In *Introduction to the Theory of Computation*, page 226?28. PWS Publishing, 1997. 37

[42] L. Tierney and A. Mira. Some adaptive Monte Carlo methods for Bayesian inference. *Statistics in Medicine*, 18:2507–2515, 1999. 41, 43

[43] P. H. Peskun. Optimum Monte Carlo sampling using markov chains. *Biometrika*, 60:607–612, 1973. 41

[44] S. EL Adlouni, A. Favre, and B. Bobee. Comparison of methodologies to assess the convergence of Markov chain Monte Carlo methods. *Computational statistics and data analysis*, 50(10):2685–2701, 2006. 49

[45] S. G. Giakoumatos, I. D. Vrontos, P. Dellaportas, and D. N. Politis. Markov Chain Monte Carlo convergence diagnostic using subsampling. *Journal of Computational and Graphical Statistics*, 8(3):431–451, September 1999. 49, 61, 62

[46] M. K. Cowles and B. P. Carlin. Markov Chain Monte Carlo convergence diagnostics: A comparative review. *Journal of the American Statistical Association*, 91:883–904, 1996. 50, 56, 59

[47] K. Mengersen, S. Knight, and C. P. Robert. MCMC: how do we know when to stop? Technical report, Bulletin of the International Statistical Institute, 1999. 52^{nd} Session, Tome LVIII, Finland. 50

[48] A. Gelman and D. B. Rubin. Inference from iterative simulation using multiple sequences. *Statistical Science*, 7(4):457–472, 1992. 51, 66, 124

[49] A. Gelman. Interdisciplinary statistics. In W. R. Gilks, S. Richardson, and D. Spiegelhalter, editors, *Markov Chain Monte Carlo in Practice*. Chapman and Hall/CRC, 1995. 51

[50] A. E. Raftery and S. M. Lewis. The number of iterations, convergence diagnostics and generic Metropolis algorithms. In W. R. Gilks and D. J. Spiegelhalter, editors, *In Practical Markov Chain Monte Carlo*, pages 115–130. Chapman and Hall, 1995. 54, 55

[51] J. Geweke. Evaluating the accuracy of sampling-based approaches to the calculation of posterior moments. In J. M. Bernardo, J. Berger, A. P. Dawid, and A. F. M. Smith, editors, *Bayesian Statistics*, pages 169–193. Oxford: Oxford University Press, 1992. 55

[52] L. Schruben, H. Singh, and L. Tierney. Optimal tests for initialization bias in simulation output. *Operations Research*, 31:1167–1178, 1983. 56

[53] B. Yu and P. Mykland. Looking at Markov samplers through cusum path plots: a simple diagnostic idea. *Statistics and Computing*, 8:275–286, 1998. 60

[54] S. P. Brooks. Quantitative convergence diagnosis for MCMC via CUSUMS. *Statistics and Computing*, 8:267–274, 1998. 60

[55] A. Zellner and C. Min. Gibbs sampler convergence criteria. *Journal of the American Statistical Association*, 90(431):921–927, September 1995. 63

[56] A. Philippe and C. P. Robert. Riemann sums for MCMC estimation and convergence monitoring. *Journal Statistics and Computing*, 11(2):1573–1375, 2001. 65

[57] S. Richardson and P. J. Green. On bayesian analysis of mixtures with an unknown number of components. *Journal of the Royal Statistical Society, Series B*, 59:731–792, 1997. 66

[58] S. P. Brooks and P. Giudici. MCMC convergence assessment via two-way ANOVA. *Journal of Computationaland Graphical Statistics*, 9:266–285, 2000. 66

[59] S. Brooks and P. Giudici. Convergence assessment for reversible jump MCMC simulations. *Bayesian Statistics*, 6:733–742, 1999. 66, 123, 134

[60] S. P. Brooks, P. Giudici, and A. Philippe. Nonparametric convergence assessment for mcmc model selection. *Journal of Computational and Graphical Statistics*, 12(1):1–22, 2003. 67

[61] A. M. Wallace, J. Ye, N. J. Krichel, and et al. Full waveform analysis for long-range 3d imaging laser radar. *EURASIP Journal on Advances in Signal Processing*, 2010:1–13, 2010. 83

[62] G. E. Moore. Cramming more components onto integrated circuits. *Electronics*, 38(8), April 1965. 85

[63] M. Flynn. Some computer organizations and their effectiveness. *IEEE Trans. Comput.*, C-21(948), 1972. 86

[64] J. Backus. Can programming be liberated from the von neumann style? a functional style and its algebra of programs. *Communications of the ACM*, 21(8):613–641, 1978. 87

[65] Y. Aoyama and J. Nakano. Rs/6000 sp: Practical mpi programming. *ITSO*, 1999. 91

[66] D. J. Wilkinson. *Handbook of parallel computing and statistics*, chapter 16: Parallel Bayesian computation, pages 477–508. CRC Press, 2006. 93, 101

[67] J. S. Rosenthal. Parallel computing and Monte Carlo algorithms. *Far East Journal of Theoretical Statistics*, 4:207–236, 2000. 94, 95, 96

[68] A. E. Brockwell. Parallel Markov chain Monte Carlo simulation by pre-fetching. *Journal of Computational and Graphical Statistics*, 15:246–261, 2006. 95, 101, 104

[69] C. J. Geyer. Practical Markov Chain Monte Carlo. *Stat. Sci.*, 7(4):473–483, 1992. 96

[70] A. Gelman and D. B. Rubin. Inference from iterative simulation using multiple sequences. *Stat. Sci.*, 7(4):457–472, 1992. 96

[71] J. N. Corcoran and R. L. Tweedie. Perfect sampling from independent Metropolis-Hastings chains. *Journal of Statistical Planning and Inference*, 104(2):297–314, June 2002. 96

[72] G. Casella, M. Lavine M, and C. P. Robert. Explaining the perfect sampler. *The American Statistician*, 55(4):299–305, November 2001. 96

[73] J. A. Fill. An interruptible algorithm for perfect sampling via Markov chains. *Annual ACM Symposium on Theory of Computing, Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 688 – 695, 1997. 96

[74] M. Luescher. A portable high-quality random number generator for lattice field theory simulations. *Comput.Phys.Commun.*, 79:100–110, 1994. 98

[75] J.L.Leva. A fast normal random number generator. *ACM Transactions on Mathematical Software (TOMS)*, 18(4):449–453, December 1992. 98

[76] G. Amdahl. Validity of the single processor approach to achieving large-scale computing capabilities. *AFIPS Conference Proceedings*, 30:483–485, 1967. 98

[77] M. A. Crane and A. J. Lemoine. *An Introduction to the Regenerative Method for Simulation Analysis*, volume 4 of *Lecture Notes in Control and Information Sciences*. Springer-Verlag, 1977. 100

[78] A. E. Brockwell and J. B. Kadane. Identification of regeneration times in MCMC Simulation, with application to adaptive schemes. *Journal of Computational and Graphical Statistics*, 14(2):436–458, June 2005. 100

[79] P. Mykland, L. Tierney, and B. Yu. Regeneration in Markov chain samplers. *Journal of the American Statistical Association*, 90:233–241, 1995. 100, 101

[80] G. L. Jones and J. P. Hobert. Honest exploration of intractable probability distributions via Markov chain Monte Carlo. *Statistical Science*, 16(4):312–334, 2001. 100

[81] C. P. Robert. Convergence control methods for Markov chain Monte Carlo algorithms. *Statistical Science*, 10:230–253, 1995. 100

[82] E. Nummelin. A splitting technique for Harris recurrent Markov chains. *Z. Wahrscheinlichkeitstheor. Verw. Geb.*, 43:309–318, 1978. 101

[83] D. Chauveau and P. Vandekerkhove. Improving convergence of the Hastings-Metropolis algorithm with an adaptive proposal. *Scandinavian Journal of Statistics*, 29(1), 2002. 101

[84] G. O. Roberts and S. K.Sahu. Updating schemes, correlation structure, blocking and parameterisation for Gibbs sampler. *Journal of Royal Statistical Society B*, 59(2):291–317, 1997. 101

[85] M. Whiley and S. P. Wilson. Parallel algorithms for Markov chain Monte Carlo methods in latent spatial Gaussian models. *Statistics and Computing*, pages 171–179, 2004. 102, 103

[86] O. D. Robles and S. P. Wilson. A set partitioning algorithm for use with a parallel approach to MCMC for latent gaussian spatial models. Technical report 02/01, Department of Statistics, Trinity College Dublin, 2002. 102

[87] R. Ren and G. Orkoulas. Parallel Markov chain Monte Carlo simulations. *Journal of Chemical Physics*, 126:211102, 2007. 102, 103, 104

[88] R. Ren and G. Orkoulas. Acceleration of Markov chain Monte Carlo simulations through sequential updating. *Journal of Chemical Physics*, 124(6):064109, 2006. 103

[89] X. Feng, D. A. Buell, J. R. Rosea, and P. J. Waddellb. Parallel algorithms for Bayesian phylogenetic inference. *Journal of Parallel and Distributed Computing*, 63(7-8):707–718, July-August 2003. 108, 114, 115

[90] C. J. Geyer. Markov chain Monte Carlo maximum likelihood. *Computing science and statistics: Procedings of the* 23^{rd} *symposium on the interface*, pages 156–163, 1991. 108

[91] J. P. Huelsenbeck and F. Ronquist. MRBAYES: Bayesian inference of phylogenetic trees. *Bioinformatics*, 17:754–755, 2001. 108

[92] W. R. Gilks and G. O. Roberts. Markov chain Monte Carlo in practice. pages 89–114. Chapman and Hall/CRC, London, 1996. 108

[93] G. Altekar, S. Dwarkadas, J. P. Huelsenbeck, and F. Ronquist. Parallel Metropolis coupled Markov chain Monte Carlo for Bayesian phylogenetic inference. *Bioinformatics*, 12(20):407–415, Febrary 2004. 108, 109, 115, 134

[94] J. Weare. Efficient Monte Carlo sampling by parallel marginalization. *Proc. Nat. Acad. Sc. USA*, 104:12657–12662, 2007. 110, 111, 113

[95] J. Liu. *Monte Carlo Strategies in Scientific Computing*. Springer, Burlin, October 2002. 110

[96] K. Binder and D. Heermann. *Monte Carlo Simulation in Statistical Physics*. Springer, Burlin, 2002. 110

[97] J. Felsenstein. Evolutionary trees from dna sequences: A maximum likelihood approach. *rnal of Molecular Evolution*, 17(6):368–376, 1981. 114

[98] D. J. Hand, F. Daly, A. D. Lunn, K. J. McConway, and et al. *Handbook for small data sets*. Chapman and Hall, London, 1994. 116

[99] S. Chib. Marginal likelihood from the Gibbs output. *Journal of the American Statistical Association*, 90(432):1313–1321, December 1995. 122

[100] S. Chib and I. Jeliazkov. Marginal likelihood from the Metropolis-Hastings output. *Journal of the American Statistical Association*, 96(453):270–281, March 2001. 122

[101] G. O. Roberts, A. Gelman, and W. R. Gilks. Weak convergence and optimal scaling of random walk metropolis algorithms. *Ann. Appl. Probab.*, 7:110–120, 1997. 181